COURS DE BASE DE DONNéES

Niveau: LF 2 INFO

Dr. Nassira ACHICH

achichnassira@gmail.com

PLAN DU COURS

- **Chapitre 1** : Rappel: Concepts fondamentaux
  - Définition d'une BD
  - Définition d'un SGBD
  - Présentation du SGBD Oracle
- **Chapitre 2** : Langage de base d'Oracle
  - Langage de définition de données : LDD
  - Langage de manipulation de données : LMD
  - Langage de contrôle de données : LCD
- **Chapitre 3** : Langage procédural : PL/SQL
  - Introduction à PL/SQL
  - Les structures de contrôle
  - Interaction avec Oracle

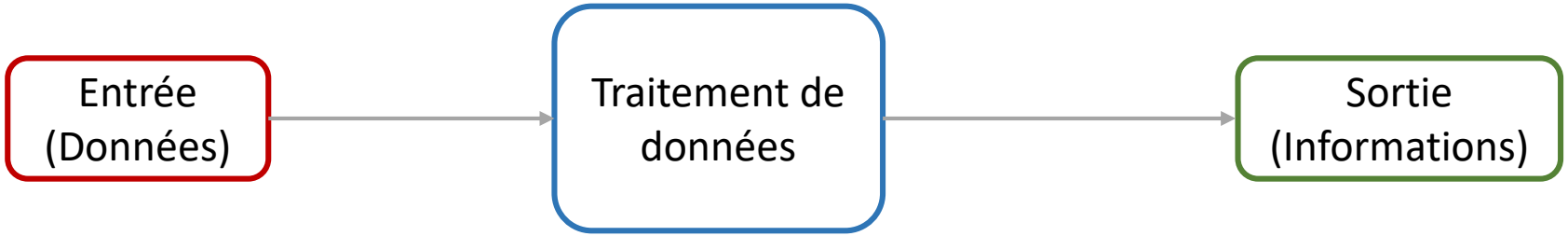
CHAPITRE 1

RAPPEL: CONCEPTS FONDAMENTAUX

QU'EST-CE QU'UNE DONNéE / INFORMATION?

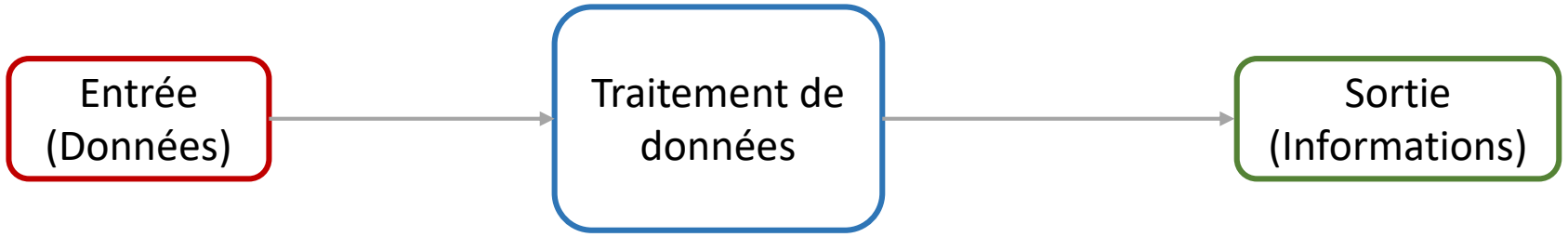
Données Vs Information ?

- **Définition : Les données** sont des entrées brutes, non analysées, non organisées, non liées, utilisées pour obtenir des informations après l'analyse.
- **L'information** est généralement le résultat traité des données. Plus spécifiquement, il est dérivé de données. L'information est un concept et peut être utilisée dans de nombreux domaines.
- Les données peuvent prendre la forme de chiffres, de caractères, de symboles ou même d'images.



Information = Données + Signification

QU'EST-CE QU'UNE DONNéE / INFORMATION?



Information = Données + Signification

Données Vs Information ?

- **Les données** sont utilisées comme entrées, qui doivent être traitées et organisées d'une manière particulière pour générer une sortie, c'est-à-dire **des informations.**
- **Les données** constituent une unité unique qui contient des faits et des chiffres bruts. En revanche, **l'information est la collecte de données utiles, capables de fournir des connaissances ou des informations sur une manière particulière.**

NAISSANCE DES BASES DE DONNéES

- Les bases de données sont apparues suite à:
  - Une évolution des entreprises
  - Une évolution du matériel
  - Une évolution des logiciels

OBJECTIFS DES BASES DE DONNéES

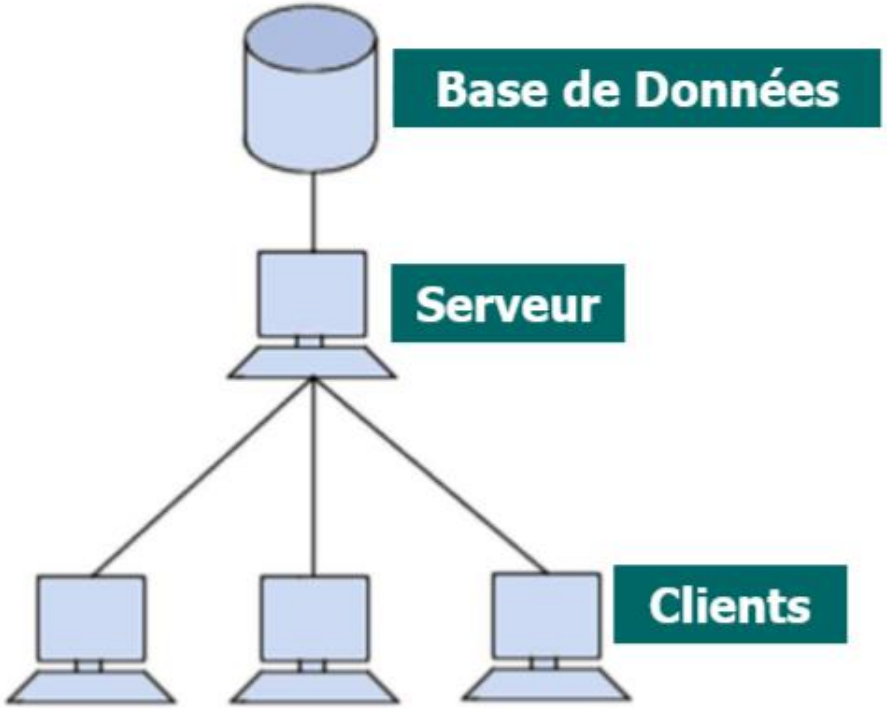
- L'utilisation d'une BD permet de remédier aux inconvénients de l'approche par fichiers.

- Centraliser l'information:
  - Elimination de la redondance des données;
  - partage les données entre les utilisateurs.
  - Intégration des données
  - Faciliter les opérations de mise à jours.

- Indépendance entre les programmes et les données;

DéFINITION D'UNE BASE DE DONNéES

- **Une base de données** est un ensemble de **données structurées enregistrées** sur un support permanent permettant leur stockage, leur manipulation et leur exploitation de manière efficace et sécurisée et accessible à plusieurs utilisateurs en même temps. Elle est gérée par un **Système de Gestion de Base de Données (SGBD)**.

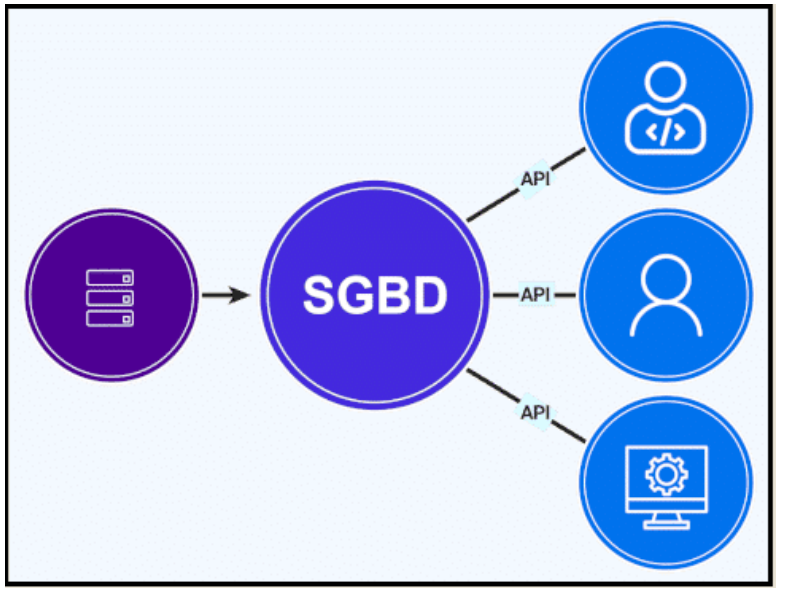


UTILISATION DES BD

- **Les smartphones**
  - Enregistrer les paramètres des applications
  - Le stockage du carnet d'adresse
- **Les navigateurs**
  - L'enregistrement des mots de passes
- **Les réseaux sociaux**
  - Stockage des paramètres des comptes
  - Stockage des recherches effectuées
  - Stockages des photos, des vidéos
- **Applications informatiques au sein des organisations**
  - Gestion des personnels, étudiants, cours, inscriptions,... de la ISIMM
  - Gestion des comptes clients d'une banque

SYSTèME DE GESTION DE BASE DE DONNéES (SGBD)

- **Un Système de Gestion de Bases de Données (SGBD)** est un logiciel qui permet de :
  - **Créer** une base de données,
  - **Manipuler** les données qu'elle contient,
  - **Contrôler** l'accès et la sécurité des données
- Un **SGBD** est un logiciel qui gère l'ensemble des fichiers constituant la BD.
- Exemples de SGBD : FoxPro, Access (Microsoft), Paradox(Borland), Oracle, Ingres, Informix, Sybase, DB2 (IBM)



OBJECTIFS D'UN SGBD

- **Non redondance des données:** Réduire au maximum l'espace mémoire utilisé pour stocker les informations,
- **Cohérence des données** : Le SGBD doit garantir un état cohérent de la BD et ce en définissant un ensemble de contraintes d'intégrités qui doit vérifier l'ensemble des informations de la BD. Exemple : age>0 ; 0<note<20.
- **Manipulations des données par des non informaticiens**: plus besoin de programmer pour accéder aux données
- **Partageabilité des données** : Il s'agit de permettre à plusieurs utilisateurs d'accéder aux mêmes données au même moment

OBJECTIFS D'UN SGBD

- **Sécurité de fonctionnement**: Le SGBD doit garantir en cas de panne logicielle ou matérielle de garder la BD dans état cohérent,

- **Sécurité des données** : Les données doivent pouvoir être protégées contre les accès non autorisés. Pour cela, il faut pouvoir associer à chaque utilisateur des droits d'accès aux données.

FONCTIONS PRINCIPALES D'UN SGBD

Les trois fonctions principales d'un SGBD sont regroupées autour de trois types de langages :

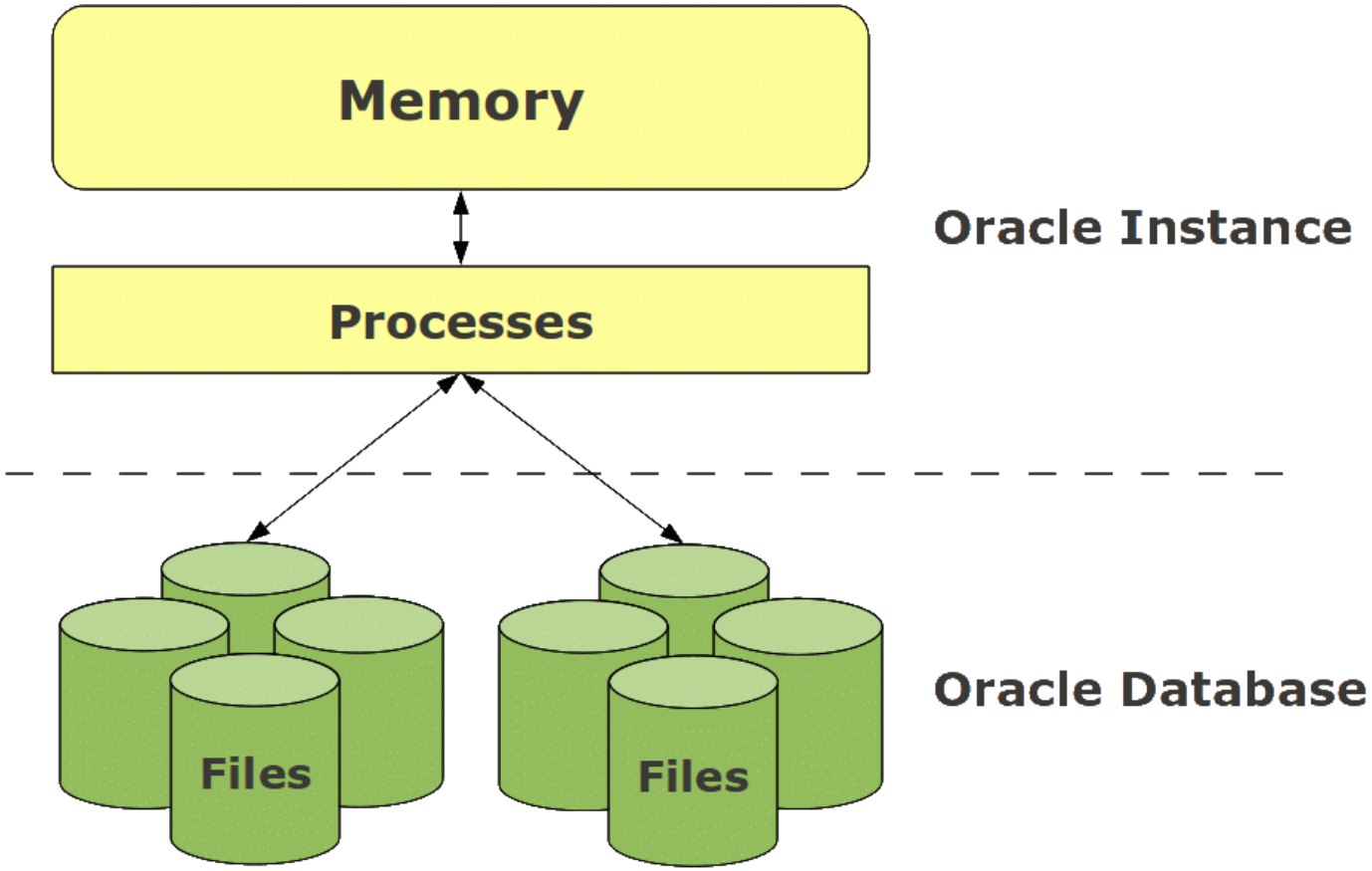
- LDD – Langage de Définition de Données
  - Sert à créer et modifier la structure d'une base de données.
  - Exemples : création de tables, de vues, de schémas, etc.
  - En SQL : CREATE, ALTER, DROP.
- LMD – Langage de Manipulation de Données
  - Permet de manipuler le contenu des tables.
  - Actions : ajouter, modifier, supprimer ou interroger des données.
  - En SQL : SELECT, INSERT, UPDATE, DELETE.
- LCD – Langage de Contrôle de Données
  - Sert à contrôler les permissions d'accès aux données.
  - En SQL : GRANT, REVOKE.
  - Permet de gérer la sécurité et les droits des utilisateurs.

ORACLE

- Oracle est un SGBDR (Système de Gestion de Bases de Données **relationnel**) édité par la société Oracle Corporation (http://www.oracle.com), leader mondial des bases de données.

- Sa fonction principale est de gérer d'une façon intégrée l'ensemble de données d'une entreprise et de les rendre accessibles à un nombre important d'utilisateurs en garantissant leur sécurité, leur cohérence et leur intégrité.

ARCHITECTURE D'ORACLE



LANGAGES UTILISéS DANS ORACLE

- Oracle utilise principalement SQL et PL/SQL (son langage procédural propre).
  - SQL dans Oracle :
    - Création de structures (LDD)
    - Manipulation des données (LMD)
    - Contrôle d'accès (LCD)
  - PL/SQL :
    - Langage procédural intégré à Oracle (boucles, conditions, procédures, fonctions…)
    - Plus puissant que le SQL standard pour la logique métier.

SQL : STRUCTURED QUERY LANGUAGE

- SQL est un langage de requête utilisé pour interagir avec les bases de données relationnelles. Il permet de créer, modifier, interroger, et gérer les données.. Créé en 1970 par IBM.

- caractéristiques de SQL
  - **Normalisation** : SQL implémente le modèle relationnel.
  - **Standard** : Du fait de cette normalisation, la plupart des éditeurs de SGBDR intègrent SQL à leurs produits (Oracle, Informix, Sybase, etc.). Ainsi, les données, requêtes et applications sont assez facilement portables d'une base à une autre.

SQL : STRUCTURED QUERY LANGUAGE

➢caractéristiques de SQL

- **Non procédural :** SQL est un langage de requêtes qui permet à l'utilisateur de demander un résultat sans se préoccuper des moyens techniques pour trouver ce résultat
- **Universel** : SQL peut être utilisé à tous les niveaux dans la gestion d'une BDR :
  - Langage de Définition de Données **LDD**,
  - Langage de Manipulation de Données **LMD**,
  - Langage de Contrôle de Données **LCD**.

SQL : STRUCTURED QUERY LANGUAGE

Type de langage	Nom	Rôle principal	Commandes/Exemples
Langage de Définition de Données	LDD (Data Definition Language)	Décrit la structure de la base de données : tables, vues, attributs, index.	CREATE, ALTER, DROP, TRUNCATE
Langage de Manipulation de Données	LMD (Data Manipulation Language)	Permet la manipulation des données dans les tables et vues.	SELECT, INSERT, DELETE, UPDATE
Langage de Contrôle de Données	LCD (Data Control Language)	Gère les transactions et les droits d'accès aux données.	COMMIT, ROLLBACK, GRANT, REVOKE

LE MODèLE RELATIONNEL

- **Le modèle relationnel** est un **modèle logique de représentation des données** sous forme de relations (tables), proposé par *Edgar F. Codd* en 1970.
- Il est à la base **des Systèmes de Gestion de Base de Données Relationnels** (SGBDR) comme Oracle, MySQL, PostgreSQL, SQL Server, etc.

LE MODèLE RELATIONNEL

Terme	Définition
Relation	un ensemble d'enregistrements (Table à deux dimensions (lignes & colonnes))
Attribut	Colonne d'une table
Tuple	Ligne d'une table (un enregistrement)
Domaine	un ensemble de valeurs caractérisées par un nom. (Ex. : Domaine (couleurs) = {bleu, rouge, blanc, …} Domaine (noms) = {ali, salah, …}
Clé primaire	Identifie de façon unique chaque tuple
Clé étrangère	Attribut qui fait référence à une clé primaire d'une autre table
Degré d'une relation	Le nombre de colonne (domaines) dans une relation

LE MODèLE RELATIONNEL

Relation PRODUIT

Enregistrement

NUM_PDT	DES_PDT	COUL_PDT
P1	D1	C1
P2	D2	C2

Domaine

Degré de PRODUIT = 3

LE MODèLE RELATIONNEL

Caractéristiques des relations

- **Attribut** : nom d'une colonne d'une relation.
  - Ex. : NUM_PDT, DES_PDT, COUL_PDT: attributs de la relation PRODUIT.
- **Schéma de relation** : nom de la relation suivi de la liste des attributs et de la définition de leurs domaines.
  - Ex. : PRODUIT (NUM_PDT, DES_PDT, COUL_PDT)
- **Clé primaire** : un attribut (ou plusieurs) permettant d'identifier d'une façon unique un tuple d'une relation. Cet attribut doit avoir toutes ses valeurs différentes dans la relation R.
  - Ex. : PRODUIT (NUM_PDT, DES_PDT, COUL_PDT)
  - ETUDIANT (NUM_ET, NOM_ET, DATNAIS_ET, ADR_ET)

LE MODèLE RELATIONNEL

➢Caractéristiques des relations

➢ **Clé étrangère** : soit la relation R1 (A, B, …, S, …).

On dit que S est une clé étrangère de R1 s'il y a une relation R2 ayant pour clé primaire S.

➢ Ex. : PRODUIT (NUM_PDT, DES_PDT, COUL_PDT, #NUM_MAG)

MAGASIN (NUM_MAG, ADR_MAG, TEL_MAG)

➢Les contraintes d'intégrité

➢ Les contraintes d'intégrité sont des règles imposées au niveau de la base de données pour assurer **la validité, la cohérence et la fiabilité des données**.

➢ Elles sont définies lors de la création des tables et sont automatiquement appliquées par le SGBD.

LE MODèLE RELATIONNEL

Type	Description	Exemple
Unicité de la clé	Identifie chaque enregistrement de manière unique	PRIMARY KEY
CI individuelles	Sur un seul champ	CHECK (salaire BETWEEN 200 AND 10000)
CI intra-relationnelles	Entre plusieurs champs de la même table	salaire ≤ 2 × moyenne(salaire)
CI inter-relationnelles	Entre des champs de tables différentes	FOREIGN KEY, qte_stk ≥ qte_cde
Contraintes dynamiques	Lors d'une mise à jour des données (transitions d'état)	nouveau_salaire ≥ ancien_salaire (trigger)

CHAPITRE 2

LANGAGE DE BASE D'ORACLE : LANGAGE DE DÉFINITION DE DONNÉES : LDD

QU'EST-CE QUE LE LDD ?

- La définition de données dans SQL permet la définition des objets manipulés par le SGBD.
- Le Langage de Définition de Données permet de créer, modifier et supprimer la structure d'une base de données.
- Sert à définir les tables, colonnes, types de données, contraintes.
- Les objets : table, vue, index.
- Les commandes du LDD sont :
  - **CREATE** : création des objets.
  - **ALTER** : modification de la structure des objets.
  - **DROP** : suppression des objets.

CRéATION DES TABLES

➢ Syntaxe de Création des tables: CREATE

➢ Celle-ci consiste à définir son nom, les colonnes qui la composent et leurs types.

Elle se fait avec la commande : **CREATE TABLE**

```
CREATE TABLE nom_table
(col 1 type [(taille)] [default ValeurParDefaut] [null / not null] [contrainte de colonne] ,
Col 2 type [(taille)] [default ValeurParDefaut] [null / not null] [contrainte de colonne],
….
Col n type [(taille)] [default ValeurParDefaut] [null / not null] [contrainte de colonne]
[contrainte de tables]);
```

CRéATION DES TABLES

➢ Syntaxe de Création des tables: CREATE

```
CREATE TABLE nom_table
(col 1 type [(taille)] [default ValeurParDefaut] [null / not null] [contrainte de colonne] ,
Col 2 type [(taille)] [default ValeurParDefaut] [null / not null] [contrainte de colonne],
….
Col n type [(taille)] [default ValeurParDefaut] [null / not null] [contrainte de colonne]
[contrainte de tables]);
```

- La **taille** indique la valeur maximale de la longueur du champ
- Les **types** de données possibles sont :
  - Numériques : INT : entier (ex: 42) / DECIMAL(p,s) : nombre décimal avec précision / FLOAT : nombre à virgule flottante
  - Caractères : CHAR(n) : chaîne fixe de longueur n / VARCHAR(n) : chaîne variable jusqu'à n caractères
  - Date/Heure : DATE : date (année-mois-jour) / TIMESTAMP : date et heure
  - Autres :BOOLEAN : vrai/faux BLOB : données binaires

CRéATION DES TABLES

➢ **Exemple 1:** *Ici on crée une table "Etudiants" avec un identifiant unique.*

```
CREATE TABLE Etudiants (
    id INT PRIMARY KEY,          -- Identifiant unique
    nom VARCHAR(50),             -- Nom de l'étudiant
    age INT,                     -- Âge
    ville VARCHAR(50)            -- Ville de résidence
);
```

CRéATION DES TABLES

➢ **Exemple 2:** *Créer la table Produit ayant comme schéma :*

Produit (Numprod, Desprod, Couleur, Poids, Qte_stk, Qte_seuil, Prix) Avec :

Numprod : de type numérique de taille 6,

Desprod : de type caractère variable de taille 15,

Couleur : de type caractère sur une position,

Poids : de type numérique sur huit positions dont trois chiffre après la virgule,

Qte_stk : de type numérique sur sept positions dont trois chiffres après la virgule,

Qte_seuil : de type numérique sur sept positions dont trois chiffres après la virgule,

Prix : de type numérique sur dix positions dont sept chiffres avant la virgule.

CRéATION DES TABLES

➢ **Exemple 2:** *Solution*

```
CREATE TABLE Produit  (
Numprod number(6) not null ,
Desprod varchar(15),
Couleur char,
Poids number(8,3),
Qte_stk number(7,3),
Qte_seuil number(7,3),
Prix number(10,3)
);
```

CRéATION DES TABLES

➢ Définition des contraintes

- Une contrainte d'intégrité est une règle qui définit la cohérence d'une donnée ou d'un ensemble de données de la base de données.
  - **NOT NULL** : interdit la valeur NULL (champ obligatoire).
  - **PRIMARY KEY** : identifie de façon unique chaque ligne.
  - **UNIQUE** : impose l'unicité dans une colonne.
  - **CHECK** : impose une condition (ex : salaire positif).
  - **FOREIGN KEY** : clé étrangère, assure la cohérence entre tables.

CRéATION DES TABLES

- Non nullité des valeurs d'un attribut : L'utilisateur est obligé de saisir la valeur de l'attribut. La commande est : **NOT NULL**
- Unicité de la valeur d'un attribut ou d'un groupe d'attributs : la valeur doit être unique. La commande est : **UNIQUE**
  - Exemple : On suppose que deux produits différents ne peuvent pas avoir la même désignation. La commande de création da la table produit devient :

```
CREATE TABLE Produit  (
Numprod number(6) not null,
Desprod varchar(15) unique ,
Couleur char,
Poids number(8,3),
Qte_stk number(7,3),
Qte_seuil number(7,3),
Prix number(10,3)
);
```

CRéATION DES TABLES

- **Clé primaire (un attribut ou un groupe)** : indique que l'attribut est une **clé primaire**. Elle peut être définie comme **contrainte de table** ou comme **contrainte de colonne**.
  - Clé primaire comme **contrainte de table** selon la syntaxe :

    CONSTRAINT nom_contrainte PRIMARY KEY(att1, att2,…, attn) ;

  - Clé primaire comme **contrainte de colonne** : en ajoutant devant la colonne clé primaire **Primary Key**.

- Dans le cas de clé primaire multiple, la clé primaire doit être créée comme contrainte de table.

CRéATION DES TABLES

➢ **Exemple 1 :** *Clé primaire comme contrainte de table*

```
CREATE TABLE Produit  (
Numprod number(6) not null,
Desprod varchar(15) unique ,
Couleur char,
Poids number(8,3),
Qte_stk number(7,3),
Qte_seuil number(7,3),
Prix number(10,3),
constraint PK_Produit primary key (NumProd)
);
```

CRéATION DES TABLES

➢ **Exemple 2 :** *Clé primaire comme contrainte de colonne*

```
CREATE TABLE Produit  (
Numprod number(6) primary key ,
Desprod varchar(15) unique ,
Couleur char,
Poids number(8,3),
Qte_stk number(7,3),
Qte_seuil number(7,3),
Prix number(10,3)
);
```

CRéATION DES TABLES

- **Exemple 3 :** *Clé primaire multiple*
  - LigneCommande (NumCde, NumProd, QteCde)

```
CREATE TABLE LigneCommande
(Num_cde number(8),
NumProd number(6),
QteCde number(7,3),
constraint pk_LigneCde primary key (NumCde, NumProd)
) ;
```

CRéATION DES TABLES

- **Clé étrangère** : lorsque la clé primaire figure dans une autre table en tant qu'un attribut non clé. La clé étrangère peut être définie comme contrainte de table ou comme contrainte de colonne.
  - Clé étrangère comme contrainte de table selon la syntaxe :

    CONSTRAINT nom_contrainte FOREIGN KEY(nom_att) references nom_table(nom_att);

- Clé étrangère comme contrainte de colonne : en ajoutant devant la colonne clé étrangère

  references nom_table(nom_att),

- Exemple: Si on considère le schéma suivant :

MAGASIN(NumMag, Adresse, Surface)

PRODUIT(NumProd, DesProd, Couleur, Poids, Qte_Stk, #CodMag)

CRéATION DES TABLES

MAGASIN(NumMag, Adresse, Surface)

PRODUIT(NumProd, DesProd, Couleur, Poids, Qte_Stk, #CodMag)

- La commande pour la création de la table Magasin étant :

Create Table Magasin(

NumMag number(6) primary Key,

Adresse varchar(30),

Surface number(7,3));

CRéATION DES TABLES

MAGASIN(NumMag, Adresse, Surface)

PRODUIT(NumProd, DesProd, Couleur, Poids, Qte_Stk, #CodMag)

- La commande pour la création de la table Produit peut être écrite de deux façons:
- Solution1

```
CREATE TABLE Produit
(Numprod number(6) primary key,
Desprod varchar(15),
Couleur char,
Poids number(8,3),
Qte_stk number(7,3),
Qte_seuil number(7,3),
Prix number(10,3),
CodMag number(6),
Constraint FK_Produit Foreing Key (CodMag) references Magasin(NumMag));
```

CRéATION DES TABLES

MAGASIN(NumMag, Adresse, Surface)

PRODUIT(NumProd, DesProd, Couleur, Poids, Qte_Stk, #CodMag)

- La commande pour la création de la table Produit peut être écrite de deux façons:
- Solution2

```
CREATE TABLE Produit
(Numprod number(6) primary key,
Desprod varchar(15),
Couleur char,
Poids number(8,3),
Qte_stk number(7,3),
Qte_seuil number(7,3),
Prix number(10,3),
CodMag number(6) references Magasin(NumMag));
```

CRéATION DES TABLES

➢ Contrainte de valeur avec la clause **CHECK :**

Permet de limiter les valeurs possibles pour une colonne en vérifiant une certaine condition. Le contrôle se fera lors des insertions des données.

Constraint nom_contrainte CHECK (colonne condition)

➢ La condition sur la colonne peut utiliser :

- un opérateur de comparaison
- la clause between val1 and val2
- la clause in (liste de valeurs)

Exemple 1: On suppose que le poids d'un produit doit être positif.

La commande de création de la table Produit devient :

CRéATION DES TABLES

➢ Contrainte de valeur avec la clause **CHECK :**

Exemple1: On suppose que le poids d'un produit doit être positif. La commande de création de la table Produit devient :

```
CREATE TABLE Produit
(Numprod number(6) primary key,
Desprod varchar(15),
Couleur char,
Poids number(8,3),
Qte_stk number(7,3),
Qte_seuil number(7,3),
Prix number(10,3),
CodMag number(6) references Magasin(NumMag),
Constraint Ck1_Produit CHECK (Poids >=0));
```

CRéATION DES TABLES

➢ Contrainte de valeur avec la clause **CHECK :**

Exemple 2: On suppose que pour un produit, la quantité en stock doit être supérieure ou égale à la quantité seuil. La commande de création de la table Produit devient :

```
CREATE TABLE Produit
(Numprod number(6) primary key,
Desprod varchar(15),
Couleur char,
Poids number(8,3),
Qte_stk number(7,3),
Qte_seuil number(7,3),
Prix number(10,3),
CodMag number(6) references Magasin(NumMag),
Constraint Ck1_Produit CHECK (Poids >=0),

);
```

CRéATION DES TABLES

➢ Contrainte de valeur avec la clause **CHECK :**

Exemple 3: On suppose que la quantité d'un produit doit être comprise entre 0 et 1000. La commande de création de la table Produit devient :

```
CREATE TABLE Produit
(Numprod number(6) primary key,
Desprod varchar(15),
Couleur char,
Poids number(8,3),
Qte_stk number(7,3),
Qte_seuil number(7,3),
Prix number(10,3),
CodMag number(6) references Magasin(NumMag),
Constraint Ck1_Produit CHECK (Poids >=0),
Constraint Ck2_Produit CHECK (Qte_stk >= Qte_seuil)),

);
```

CRéATION DES TABLES

➢ Contrainte de valeur avec la clause **CHECK :**

Exemple 4: On suppose que la couleur d'un produit ne peut être que 'N', 'G', ou 'B'. La commande de création de la table Produit devient :

```
CREATE TABLE Produit
(Numprod number(6) primary key,
Desprod varchar(15),
Couleur char,
Poids number(8,3),
Qte_stk number(7,3),
Qte_seuil number(7,3),
Prix number(10,3),
CodMag number(6) references Magasin(NumMag),
Constraint Ck1_Produit CHECK (Poids >=0),
Constraint Ck2_Produit CHECK (Qte_stk >= Qte_seuil)),
Constraint Ck3_Produit CHECK (Qte_stk between 0 and 1000),

);
```

CRéATION DES TABLES

➢ **Exercice**

Compagnie(NComp, NomComp, AdrComp, Ville)

Pilote (NPil, Nom, NbHvol, #Compa)

Travail demandé: Créer les deux tables avec les contraintes nécessaires.

Les contraintes doivent être créée au niveau table.

CRéATION DES TABLES

➢ **Solution**

Compagnie(NComp, NomComp, AdrComp, Ville)

Pilote (NPil, Nom, NbHvol, #Compa)

```
CREATE Table Compagnie
(NComp char(4),
NomComp varchar2(15),
AdrComp varchar2(20),
Ville varchar2(15),
Constraint pk_Compagnie PRIMARY KEY (NComp));
```

CRéATION DES TABLES

➢ **Solution**

Compagnie(NComp, NomComp, AdrComp, Ville)

Pilote (NPil, Nom, NbHvol, #Compa)

```
CREATE Table Pilote
(NPil CHAR(6),
Nom CHAR(15) NOT NULL,
NbHvol NUMBER(7,2),
Compa CHAR (4),
Constraint pk_Pilote PRIMARY KEY (NPil),
Constraint ck_NbHvol CHECK (NbHvol BETWEEN 0 And 2000),
Constraint un_nom UNIQUE (nom),
Constraint FK_Pil_Compa FOREIGN KEY (Compa) REFERENCES Compagnie (NComp));
```

CRéATION DES TABLES

➢ **Exercice**

Université (CodeU, NomU, AdresseU, VilleU)

Professeur (CodeP, NomP, Spécialité, #CodeU)

Travail demandé: Créer les deux tables avec les contraintes nécessaires.

Les contraintes doivent être créée au niveau colonne.