

# GESTION DE DONNÉES

Les concepts fondamentaux

# Bases de données :

(notions de bases)

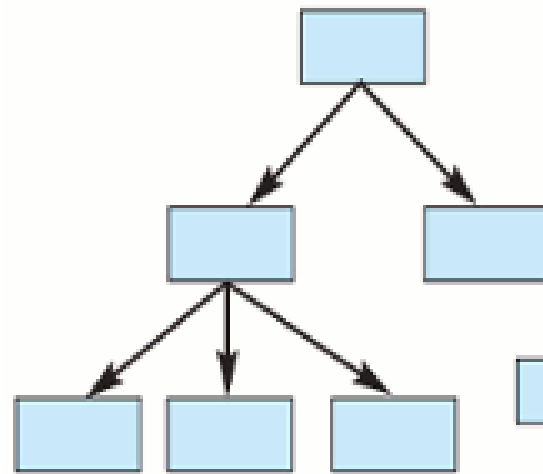
# 1. Définition d'une base de données

Une base de données est une collection de données structurées relatives à un ou plusieurs domaines du monde réel. Elle est **persistante, structurée, non redondante** et **exhaustive**.

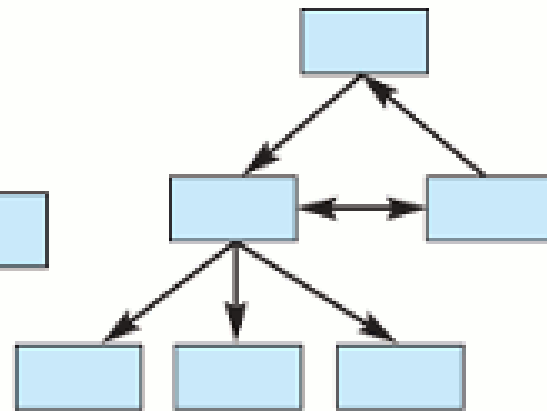
## 2. Intérêts de l'utilisation d'une base de données

- Centralisation
- Indépendance entre données et programmes  
*surtout pour éviter les différences des lagunages de programmation*
- Intégrité des données  
*ensembles des règles qui permet la cohérence des données et qui (règles) doit être tjrs vérifiés*
- Partage de données
- Intégration des liaisons entre données:  
*les données sont reliées entre eux pour donner un sens*

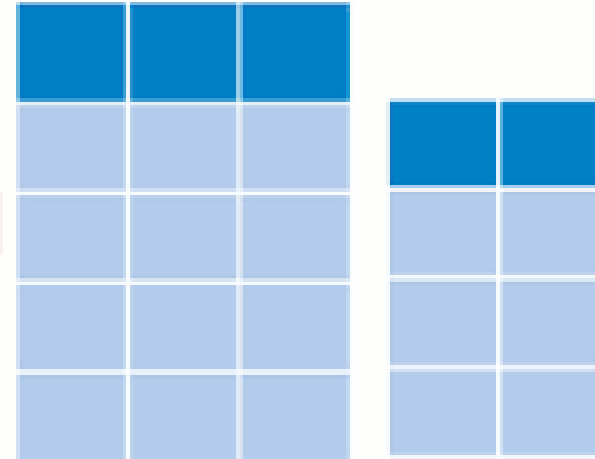
### 3. Les modèles de bases de données



**Modèle  
hiérarchique**



**Modèle réseau**



**Modèle  
relationnel**

### 3. Les modèles de bases de données

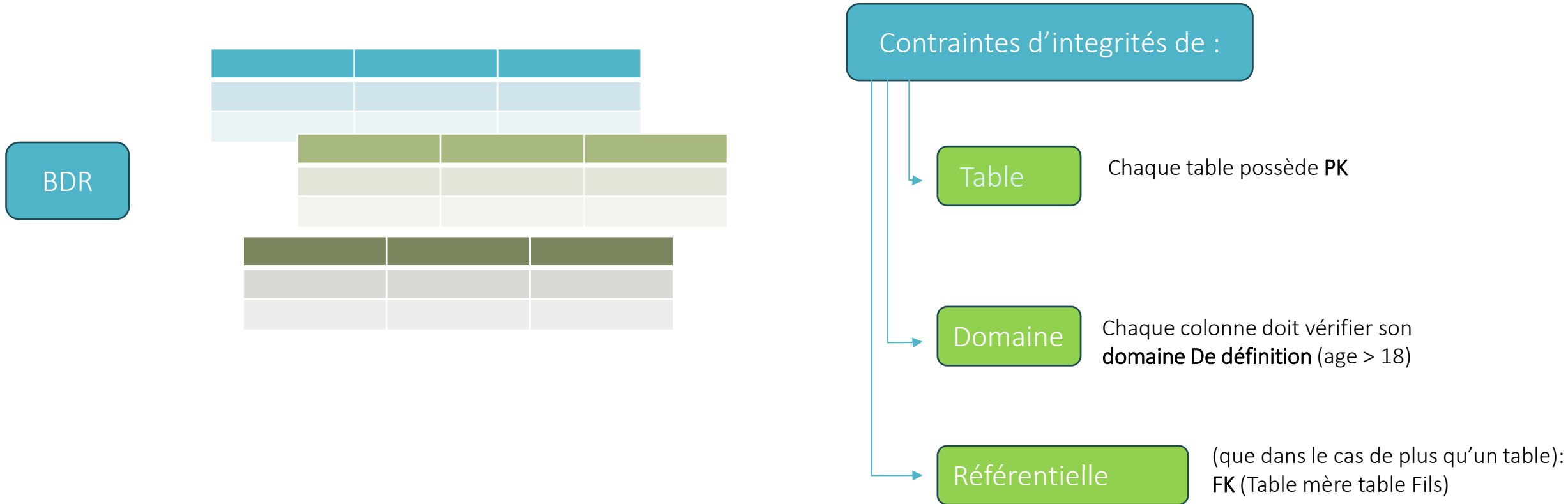
**Modèle hiérarchique** : organise les données dans une structure arborescente, où chaque enregistrement dispose d'un seul parent.

**Modèle réseau** : est une extension du modèle hiérarchique qui autorise des relations plusieurs-à-plusieurs entre des enregistrements liés.

**Modèle relationnel** : Consiste à présenter les objets et les liens à l'aide d'une structure appelée table.

**Modèle orienté objet** : Ce modèle définit une base de données comme une collection d'objets associés à des caractéristiques et des méthodes.

### 3. Modèle relationnel BDR



# Notion de Système de Gestion de Bases de Données

(SGBD)



# 1. Définition d'un SGBD

Un Système de Gestion de Base de Données (SGBD) :

est un logiciel qui permet de **créer, organiser, modifier et consulter** des données de manière efficace et sécurisée.

En gros, c'est comme une **bibliothèque numérique** où les informations sont rangées dans des "étagères" (tables), et le SGBD aide à les retrouver facilement.



## 2. Les fonctionnalités d'un SGBD

- La définition des données
- La manipulation des données
- L'intégrité des données
- La gestion des accès concurrents
- La confidentialité

# Structure d'une BDR

## Base de Données Relationnelle

(SGBD)

# 1. Notion de table (table, entité, sujet, relation)

Les données d'une **BD** sont réparties sur un ensemble de tables.

Une table est un ensemble de données relatives à un même sujet (ou *entité*) et structurées sous forme de tableau.

**Exemple** : Structure de la table Personne

Colonne (champ)

CIN	Nom	Prenom	DateNaiss
05584123	Tounsi	Ahmed	12/01/2005
02774412	Karim	Thabet	23/05/1988
05893320	Selmi	Fahmi	08/07/2008

Lignes (enregistrement)

## 2. Notion de colonne (champ, attribut, propriété)

Dans une table, une colonne correspond à une propriété élémentaire de l'objet décrit par cette table.

Une colonne est décrite par :

- *un nom,*
- *un type de données,*
- *une taille,*
- *une valeur par défaut,*
- *un indicateur de présence obligatoire*
- *et une règle indiquant les valeurs autorisées.*

### 3. Notion de ligne (enregistrement, occurrence, n-uplet)

Une ligne correspond à une occurrence du sujet représenté par la table.

**Exemple :**

*La table personne contient 3 lignes correspondant chacune à une personne.*

## 4. Notion de clé primaire

La ***clé primaire*** d'une table est une colonne ou un groupe de colonnes permettant d'identifier de façon unique chaque ligne de la table.

La connaissance de la valeur de la clé primaire permet de connaître sans ambiguïté les valeurs des autres colonnes de la table.

## 5. Notion de clé étrangère

Les différents sujets de chaque domaine sont généralement interreliés par des liens (association).

Les liens entre les sujets doivent se retrouver dans la base de données.



## 6. Représentation d'une BD

### la représentation textuelle

client(idClient, nom, email)

article(idArticle, nomArticle, marque, prix)

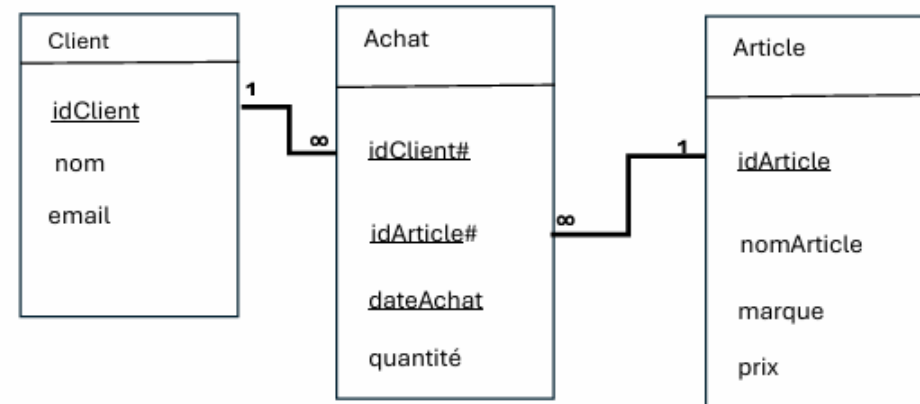
achat (idClient#, idArticle#, dateAchat, quantité)

### Remarque:

*On Adopte, pour la représentation textuelle et graphique d'une BDR, la convention suivante:*

- clé primaire soulignée
- clé étrangère suivie du symbole #

### La représentation graphique



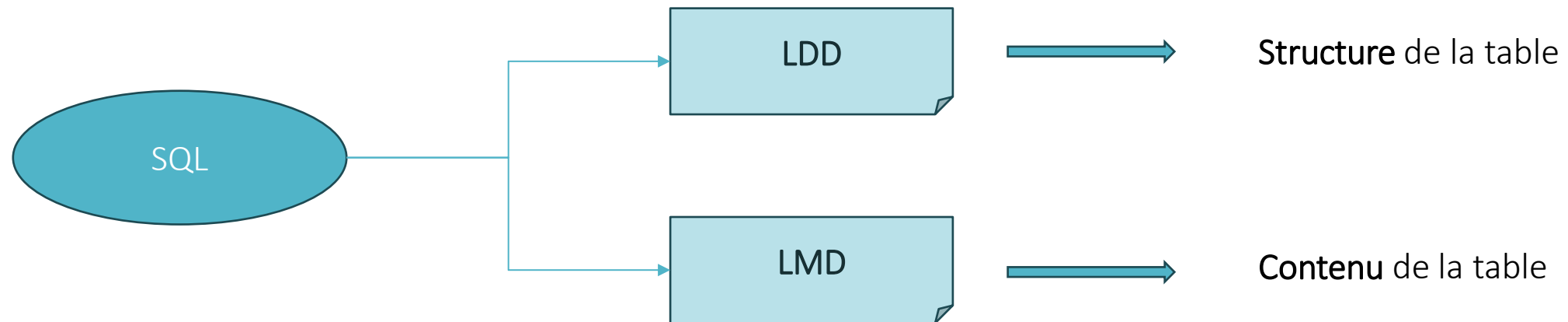
# Manipulations sur la structure d'une base de données :

(Définition des données)

# 1. Langage SQL :

SQL (***Structured Query Language***) est un langage standard utilisé pour interagir avec les bases de données relationnelles.

Il permet d'effectuer diverses opérations comme la **création**, la **modification**, la **gestion** et la requête des données stockées dans des bases de données sous forme de tables.



## 2. Définition des bases :

La commande **CREATE DATABASE** est utilisée pour créer une nouvelle base de données dans un système de gestion de bases de données (SGBD) comme MySQL

```
CREATE DATABASE Bd_Letraditionnel;
```

La commande **DROP DATABASE** est utilisée pour supprimer une base de données entière, ainsi que toutes les tables et données qu'elle contient. Cette opération est irréversible et doit être utilisée avec précaution, car elle supprime définitivement toutes les données de la base de données.

```
DROP DATABASE Bd_Letraditionnel;
```

### 3. Définition des tables :

La commande **CREATE TABLE** en SQL est utilisée pour créer une nouvelle table dans une base de données.

Elle permet de définir la structure de la table en spécifiant les colonnes, leurs types de données et d'autres contraintes comme les clés primaires, les clés étrangères, les valeurs par défaut, etc.

```
CREATE TABLE NomTable (  
    Colonne1 TypeDeDonnée (taille) [Contraintes],  
    Colonne2 TypeDeDonnée (taille) [Contraintes],  
    , [contraintes définition complète]  
);
```

## a. Les types des données en SQL :

Type SQL	Description	Exemple d'utilisation	Exemple d'utilisation
INT	Entier sur 4 octets (32 bits), utilisé pour stocker des nombres entiers.	INT : Stocke des valeurs entières comme 100, -50, 2023.	AGE INT;
DECIMAL	Nombre décimal avec une précision fixe (nombre total de chiffres) et une échelle (chiffres après la virgule).	DECIMAL(10, 2) : Nombre avec jusqu'à 10 chiffres au total, dont 2 après la virgule.	PRIX DECIMAL(10, 2);
CHAR	Chaîne de caractères de longueur fixe. Si la longueur n'est pas atteinte, la chaîne est complétée par des espaces.	CHAR(10) : Chaîne fixe de 10 caractères.	CODE CHAR(10); pour des codes de produit de longueur fixe.
VARCHAR	Chaîne de caractères de longueur variable. La taille est flexible jusqu'à une limite définie.	VARCHAR(50) : Chaîne de longueur variable jusqu'à 50 caractères.	NOM VARCHAR(50); pour stocker des noms.
TEXT	Utilisé pour de longues chaînes de caractères, comme des paragraphes ou des descriptions.	TEXT : Texte long sans limite de taille prédéfinie.	DESCRIPTION TEXT; pour des descriptions détaillées.
DATE	Représente une date au format AAAA-MM-JJ (année-mois-jour).	DATE : Stocke une date	DATE_NAISSANCE DATE; avec une valeur comme 1990-05-12.
TIME	Représente une heure au format HH:MM (heure:minute).	TIME : Stocke l'heure.	HEURE_ENTRÉE TIME; avec une valeur comme 14:30:00
DATETIME	Combine une date et une heure au format AAAA-MM-JJ HH:MM	DATETIME : Stocke une date et une heure	CREATION DATETIME; avec une valeur comme 2023-12-01 14:30:00.

# Les contraintes :

Contrainte	Description	Exemple d'utilisation
PRIMARY KEY	Assure que chaque enregistrement dans une table est unique et non nul.	ClientID INT PRIMARY KEY
NOT NULL	Empêche une colonne de contenir des valeurs nulles (obligatoire).	Nom VARCHAR(50) NOT NULL
UNIQUE	Assure que toutes les valeurs d'une colonne (ou d'un ensemble de colonnes) sont uniques dans la table.	Email VARCHAR(100) UNIQUE
DEFAULT	Définit une valeur par défaut pour une colonne si aucune valeur n'est spécifiée lors de l'insertion.	DateInscription DATE DEFAULT "1990-01-01"
REFERENCES	Déclare une clé étrangère (foreign key) qui fait référence à une colonne d'une autre table pour garantir l'intégrité référentielle.	ClientID INT REFERENCES Client(ClientID);
AUTO_INCREMENT	Incrémente automatiquement la valeur à chaque nouvelle insertion. S'applique à une colonne de type entier (INT)	id INT AUTO_INCREMENT PRIMARY KEY

## b. Contraintes définition complète: FOREIGN KEY

- ❖ La contrainte **FOREIGN KEY** (clé étrangère) est utilisée pour lier deux tables.
- ❖ Elle fait référence à une colonne ou à un ensemble de colonnes dans une autre table, généralement à une **PRIMARY KEY**.
- ❖ Elle garantit *l'intégrité référentielle* entre les tables.

*Cela signifie que la valeur de la clé étrangère dans la table enfant doit exister dans la table parent.*



## b. Contraintes définition complète: FOREIGN KEY

```
CREATE TABLE NomTable (  
    Colonne1 TypeDeDonnée (taille) [Contraintes],  
    Colonne2 TypeDeDonnée (taille) [Contraintes],  
    , FOREIGN KEY (ColonneFK) REFERENCES TableParent(ColonneParent) );
```

### Exemple :

```
CREATE TABLE Commande (  
    CommandeID INT PRIMARY KEY,  
    ClientID INT,  
    DateCommande DATE,  
    FOREIGN KEY (ClientID) REFERENCES Client(ClientID) );
```

## Remarques :

Les clauses *ON UPDATE CASCADE* et *ON DELETE CASCADE*:

sont utilisées en conjonction avec les contraintes de clé étrangère (FOREIGN KEY) dans SQL.

Elles définissent le comportement à adopter lorsqu'une ligne dans la table parent est mise à jour ou supprimée, en synchronisant automatiquement les modifications dans la table enfant.

## Exemple :

```
CREATE TABLE Clients (  
    ClientID INT PRIMARY KEY,  
    Nom VARCHAR(100)  
);
```

```
CREATE TABLE Commandes (  
    CommandeID INT PRIMARY KEY,  
    ClientID INT,  
    FOREIGN KEY (ClientID) REFERENCES Clients(ClientID)  
    ON UPDATE CASCADE ON DELETE CASCADE  
);
```

## Explication :

- ON UPDATE CASCADE :

Si l'ID d'un client dans Clients est mis à jour,

la colonne ClientID dans Commandes sera automatiquement mise à jour.

- ON DELETE CASCADE :

Si un client est supprimé de Clients,

toutes ses commandes seront également supprimées de Commandes.

## b. Contraintes définition complète: CHECK

```
CREATE TABLE NomTable (  
    Colonne1 TypeDeDonnée CHECK (condition),  
    Colonne2 TypeDeDonnée  
);
```

### Exemple :

```
CREATE TABLE Commande (  
    ProduitID INT PRIMARY KEY,  
    Nom VARCHAR(100) NOT NULL,  
    Prix DECIMAL(10, 2),  
    QuantitéStock INT CHECK (QuantitéStock >= 0)  
    -- QuantitéStock doit être supérieure ou égale à 0  
);
```

## c. Contraintes définition complète: CHECK (Opérateurs de comparaison SQL)

Opérateur	Description	Exemple SQL avec CHECK	Explication de l'exemple
=	Vérifie l'égalité entre deux valeurs	CHECK (Age = 30)	Restreint l'âge à la valeur exacte de 30.
< > ou !=	Vérifie l'inégalité (différent de)	CHECK (Age <> 30)	Interdit les enregistrements où l'âge est exactement 30.
>	Vérifie si la valeur de gauche est supérieure à celle de droite	CHECK (Age > 18)	Permet uniquement des âges supérieurs à 18.
<	Vérifie si la valeur de gauche est inférieure à celle de droite	CHECK (Age < 65)	Permet uniquement des âges inférieurs à 65.
> =	Vérifie si la valeur de gauche est supérieure ou égale à celle de droite	CHECK (Salaire >= 3000)	Garantit que le salaire est d'au moins 3000.
< =	Vérifie si la valeur de gauche est inférieure ou égale à celle de droite	CHECK (Salaire <= 10000)	Garantit que le salaire ne dépasse pas 10 000.
BETWEEN	Vérifie si une valeur est comprise entre deux valeurs	CHECK (Age BETWEEN 18 AND 65)	Restreint l'âge à être entre 18 et 65 inclus.
IN	Vérifie si une valeur est présente dans une liste de valeurs	CHECK (Statut IN ('Actif', 'Inactif'))	Limite le statut aux valeurs "Actif" ou "Inactif".

## c. Contraintes définition complète: CHECK (opérateurs Logique SQL)

Opérateur	Description	Exemple	Explication de l'exemple
AND	Renvoie vrai si toutes les conditions sont vraies	CHECK (Age >= 18 AND Age <= 65)	Limite l'âge à être entre 18 et 65.
OR	Renvoie vrai si au moins une condition est vraie	CHECK (Sexe = 'F' OR Sexe = 'M')	Permet que le sexe soit soit "F" soit "M".
NOT	Renverse le résultat d'une condition	CHECK (NOT (Age < 18))	Interdit les enregistrements où l'âge est inférieur à 18.

## c. Contraintes définition complète: CLÉ PRIMAIRE Composée

Une clé primaire composée est une clé primaire qui est définie sur plusieurs colonnes (ou champs) d'une table.

Cela signifie que la combinaison de ces colonnes doit être unique et non nulle pour chaque enregistrement, mais aucune de ces colonnes ne peut, à elle seule, identifier de manière unique une ligne dans la table.

### **Pourquoi utiliser une clé primaire composée ?**

- Une clé primaire composée est utilisée lorsque plusieurs colonnes ensemble définissent de manière unique un enregistrement, mais qu'aucune colonne prise individuellement ne le fait.
- C'est souvent utilisé dans les tables de relation pour modéliser des associations entre plusieurs tables.



## c. Contraintes définition complète: CLÉ PRIMAIRE Composée

Exemple:

Imaginons une table Inscription qui enregistre l'association entre des étudiants et des cours. Chaque étudiant peut suivre plusieurs cours, et chaque cours peut être suivi par plusieurs étudiants. Pour garantir qu'un étudiant ne s'inscrit pas plusieurs fois au même cours, nous allons utiliser une clé primaire composée des colonnes EtudiantID et CoursID.

```
CREATE TABLE Inscription (  
    EtudiantID INT,  
    CoursID INT,  
    DateInscription DATE,  
    PRIMARY KEY (EtudiantID, CoursID),  
    FOREIGN KEY (EtudiantID) REFERENCES Etudiant(EtudiantID),  
    FOREIGN KEY (CoursID) REFERENCES Cours(CoursID)  
);
```

## d. Supprimer une table :

La requête SQL DROP TABLE est utilisée pour supprimer une table existante dans une base de données, ainsi que toutes les données qu'elle contient. Une fois exécutée, cette opération est irréversible, c'est-à-dire que la table et ses données sont définitivement supprimées.

### **DROP TABLE NomTable;**

Cette commande supprime la table Client ainsi que toutes les données et les contraintes associées à cette table.

### Remarque :

*Si la table a des relations (clés étrangères) avec d'autres tables, il peut être nécessaire de désactiver ou de supprimer ces relations avant d'exécuter la commande **DROP TABLE***

# 4. Définition des colonnes :

La commande SQL **ALTER TABLE** permet de modifier la structure d'une table existante dans une base de données.

Cela inclut l'ajout, la suppression, la modification, ou le renommage de colonnes.

Opération	Syntaxe	Exemple
Ajouter une colonne	<b>ALTER TABLE NomTable</b>  ADD COLUMN NomColonne TypeDeDonnée;	ALTER TABLE Client  ADD COLUMN Email VARCHAR(100);
Supprimer une colonne	<b>ALTER TABLE NomTable</b>  DROP COLUMN NomColonne;	ALTER TABLE Client  DROP COLUMN Adresse;
Modifier le type d'une colonne	<b>ALTER TABLE NomTable</b>  ALTER COLUMN NomColonne NouveauType;	ALTER TABLE Client  ALTER COLUMN Age VARCHAR(3);
Renommer une colonne	<b>ALTER TABLE NomTable</b>  RENAME COLUMN AncienNom TO NouveauNom;	ALTER TABLE Client  RENAME COLUMN Nom TO NomClient;

# 5. Définitions des contraintes :

La commande SQL **ALTER TABLE** permet de modifier la structure d'une table en ajoutant ou supprimant des contraintes, ou en activant/désactivant des contraintes existantes.

Opération	Syntaxe	Exemple
<b>Ajouter une contrainte</b>	ALTER TABLE NomTable ADD CONSTRAINT NomContrainte TypeDeContrainte(NomColonne);	<b>ALTER TABLE</b> Commandes <b>ADD CONSTRAINT</b> fk_client FOREIGN KEY (ClientID) REFERENCES Clients(ClientID);
<b>Supprimer une contrainte</b>	ALTER TABLE NomTable DROP CONSTRAINT <i>NomContrainte</i> ;	<b>ALTER TABLE</b> Commandes <b>DROP CONSTRAINT</b> fk_client;
<b>Activer une contrainte</b>	ALTER TABLE NomTable ENABLE CONSTRAINT <i>NomContrainte</i> ;	<b>ALTER TABLE</b> Commandes <b>ENABLE CONSTRAINT</b> fk_client;
<b>Désactiver une Contrainte</b>	ALTER TABLE NomTable DISABLE CONSTRAINT <i>NomContrainte</i> ;	<b>ALTER TABLE</b> Commandes <b>DISABLE CONSTRAINT</b> fk_client;

# 5. Définitions des contraintes :

Types de contraintes courantes :

1. PRIMARY KEY : Un identifiant unique pour chaque ligne de la table.
2. FOREIGN KEY : Crée un lien entre deux tables.
3. CHECK : Impose une règle sur les valeurs qu'une colonne peut accepter.

# Manipulations sur les données d'une base de données:

(Manipulations des données)

# 1. Insertion des données :

La commande SQL **INSERT INTO** est utilisée pour insérer des nouvelles données dans une table d'une base de données.

*Vous pouvez insérer des données dans toutes les colonnes d'une table ou seulement dans certaines colonnes spécifiques.*

Action	Syntaxe	Exemple
Insérer dans toutes les colonnes	<b>INSERT INTO</b> NomTable <b>VALUES</b> (valeur1, valeur2, ..., valeurN);	<b>INSERT INTO</b> Clients <b>VALUES</b> (1,'Ahmed','ahmed@example.com');
Insérer dans certaines colonnes	<b>INSERT INTO</b> NomTable (Colonne1, Colonne2, ...) <b>VALUES</b> (valeur1, valeur2, ...);	<b>INSERT INTO</b> Clients (Nom, Email) <b>VALUES</b> ('Layla', 'layla@example.com');
Insérer plusieurs lignes	<b>INSERT INTO</b> NomTable (Colonne1, Colonne2, ...) <b>VALUES</b> (valeur1, valeur2), (valeur1, ...);	<b>INSERT INTO</b> Clients (Nom, Email) <b>VALUES</b> ('Ahmed', 'ahmed@example.com'), ('Layla', 'layla@example.com');

## 2. Suppression des données :

La commande **DELETE** permet de supprimer des enregistrements dans une table.

*Elle peut être utilisée pour supprimer toutes les lignes d'une table ou seulement certaines lignes, en fonction d'une condition spécifiée dans la clause **WHERE**.*

Action	Syntaxe	Exemple
Suppression simple	<b>DELETE FROM</b> NomTable <b>WHERE</b> condition;	<b>DELETE FROM</b> Clients <b>WHERE</b> ClientID = 1;
Suppression de tous les enregistrements	<b>DELETE FROM</b> NomTable;	<b>DELETE FROM</b> Clients;



# 3. Modifications des données :

La requête UPDATE permet de modifier les valeurs d'une table en fonction de conditions spécifiques. Sans jointure, elle ne concerne qu'une seule table.

Action	Syntaxe	Exemple
Mettre à jour une valeur	<b>UPDATE</b> table_name <b>SET</b> column_name = new_value <b>WHERE</b> condition;	Modifie le salaire de l'employé dont l'ID est 1. <b>UPDATE</b> Employes <b>SET</b> salaire = 4000 <b>WHERE</b> employe_id = 1;
Mettre à jour plusieurs colonnes	<b>UPDATE</b> table_name <b>SET</b> column1 = new_value1, column2 = new_value2 <b>WHERE</b> condition;	Met à jour le salaire et le département de l'employé avec l'ID 3. <b>UPDATE</b> Employes <b>SET</b> salaire = 4500 ,departement = 'Marketing' <b>WHERE</b> employe_id = 3;
Mettre à jour toutes les lignes ( <b>sans WHERE</b> )	<b>UPDATE</b> table_name <b>SET</b> column_name = new_value;	Modifie le salaire de tous les employés dans la table. <b>UPDATE</b> Employes <b>SET</b> salaire = 5000;
Modifier une valeur existante	<b>UPDATE</b> table_name <b>SET</b> column_name = column_name + increment_value <b>WHERE</b> condition;	Augmente le salaire des employés du département IT de 500. <b>UPDATE</b> Employes <b>SET</b> salaire = salaire + 500 <b>WHERE</b> departement = 'IT';

# 4. Sélection des données

# 3. Modifications des données :

La requête **SELECT** en SQL permet de récupérer des données dans une table de base de données.

C'est l'une des opérations les plus basiques et les plus importantes en SQL.

## Structure de base d'une requête SELECT

**SELECT** colonnes **FROM** table;

### ❖ SELECT

- Cela signifie : choisir les colonnes que tu veux afficher.
- Par exemple : `SELECT nom, age` signifie que tu veux récupérer les colonnes nom et age.
- `SELECT *` signifie sélectionner tous les colonnes.

### ❖ FROM

- C'est là où tu indiques de quelle table(s) tu veux récupérer les données. Par exemple : `FROM utilisateurs`.

# Sélection des données

(Manipulations des données)

Filtrage des résultats:

---

La clause **WHERE** en SQL est utilisée pour filtrer les enregistrements dans une requête **SELECT** en fonction d'une ou plusieurs conditions.  
Elle permet de récupérer uniquement les lignes qui répondent aux critères spécifiés.

### Syntaxe générale

**SELECT** colonne1, colonne2, ... **FROM** table **WHERE** condition;

Fonctionnement de la clause WHERE

- **SELECT** : Utilisé pour spécifier les colonnes que vous souhaitez récupérer.
- **FROM** : Indiquer la(les) table(s) à partir de laquelle les données sont extraites.
- **WHERE** : Spécifie les conditions que les lignes doivent respecter pour être incluses dans les résultats.

### Exemple:

Supposons que nous ayons une table employés comme ci-dessous :

id_employe	nom	age	salaire
1	Alice	30	50000
2	Bob	45	60000
3	Charlie	25	40000
4	David	35	55000

---

### Exemple 01:

**SELECT** nom, salaire **FROM** employes **WHERE** salaire > 50000;

### Exemple 02: (Opérateur **AND**)

**SELECT** nom, age, salaire **FROM** employes **WHERE** age > 30 **AND** salaire > 50000;

### Exemple 03: (Opérateur **OR**)

**SELECT** nom, age, salaire **FROM** employes **WHERE** age > 30 **OR** salaire > 50000;

#### Exemple 04: (Opérateur LIKE)

```
SELECT nom FROM employes WHERE nom LIKE 'A%';
```

Ici, l'opérateur **LIKE** est utilisé pour rechercher des correspondances partielles. 'A%' signifie "commence par A".

Joker	Description	Exemple de motif	Explication du motif
%	Représente zéro, un ou plusieurs caractères	C%	Correspond à toute chaîne commençant par "C"
_	Représente exactement un caractère	C_____	Correspond à toute chaîne commençant par "C" suivie de 7 autres caractères



### Exemple 05: (Opérateur IS NULL / IS NOT NULL)

En SQL, les opérateurs IS NULL et IS NOT NULL sont utilisés pour vérifier la présence ou l'absence de valeurs NULL dans une colonne. Voici leurs fonctions et des exemples d'utilisation.

Soit table appelée employes suivant :

id_employe	nom	age	salaire
1	Alice	30	50000
2	Bob	NULL	60000
3	Charlie	25	NULL
4	David	35	55000

```
SELECT nom,age FROM employes WHERE age IS NULL;
```

# Sélection des données

(Manipulations des données)

Jointure

---

En SQL, une jointure est utilisée pour combiner des lignes provenant de deux ou plusieurs tables en fonction d'une condition qui lie les tables, souvent basée sur une relation entre une clé primaire et une clé étrangère.

*Pour faire la jointure les tables sont listées dans la clause **FROM**, et la condition de jointure est définie dans la clause **WHERE**.*

### Syntaxe générale

```
SELECT colonne1, colonne2, ... FROM Table1, Table2  
WHERE Table1.colonne = Table2.colonne;
```

---

**Exemple :**

```
SELECT A.Nom, C.Produit FROM  
Clients AS cl,  
Commandes AS cmd  
WHERE  
cl.ClientID = cmd.ClientID;
```

Client

ClientID	Nom
1	Ahmed
2	Layla

Commande

CommandeID	ClientID	Produit
101	1	Ordinateur
102	2	Smartphone
103	1	Imprimante

# Sélection des données

(Manipulations des données)

Fonctions sur les  
types DATE

# Sélection des données

(Manipulations des données)

Fonctions  
d'agrégation

# Sélection des données

(Manipulations des données)

Groupement

# Sélection des données

(Manipulations des données)

Tri



# Sélection des données

(Manipulations des données)

Sous-requête