Création, modification des tables et des vues en SQL
Objectifs du chapitre :
À la fin de ce cours, l'étudiant sera capable de :

- Créer correctement des tables en SQL
- Définir des contraintes d'intégrité
- Modifier une table existante
- Supprimer une table
- Créer et utiliser des vues
- Comprendre l'intérêt des vues en entreprise

1) Rappels : qu'est-ce qu'une table en base de données ?

Une **table** représente une **relation** entre plusieurs données.
Elle est composée de :

- **Colonnes** → attributs
- **Lignes** → enregistrements (tuples)

Exemple de table Produit :

Nump	Nom	Prix	Stock
1	Clavier	2500	15
2	Souris	1200	40

2) Création d'une table en SQL – CREATE TABLE

2.1 Syntaxe générale

```
CREATE TABLE nom_table (
  colonne1 type [contraintes],
  colonne2 type [contraintes],
  ...
);
```

2.2 Types de données les plus utilisés

Type	Utilité
INTEGER	nombres entiers
REAL	nombres réels
CHAR(n)	taille fixe
VARCHAR(n)	taille variable
DATE	date

3) Les contraintes d'intégrité (le cœur de la fiabilité)

Les contraintes servent à **protéger la cohérence des données**.

3.1 Contrainte de non-nullité — NOT NULL

Empêche la valeur NULL.

```
CREATE TABLE Categorie (
  IdCat INTEGER NOT NULL,
  NomCat VARCHAR(50) NOT NULL
);
```

Impossible d'ajouter une catégorie sans nom.

3.2 Contrainte d'unicité — PRIMARY KEY et UNIQUE

Clé primaire (PRIMARY KEY)

- Identifie **de façon unique** chaque ligne
- Automatiquement NOT NULL + UNIQUE

```
CREATE TABLE Produit (
  Nump INTEGER PRIMARY KEY,
  Nom VARCHAR(50),
  Prix REAL
);
```

Contrainte UNIQUE

Email VARCHAR(100) UNIQUE

Deux utilisateurs ne peuvent pas avoir le même email.

3.3 Contrainte référentielle — FOREIGN KEY

Elle permet de **créer un lien entre deux tables**.

Exemple :

Table Categorie :

```
CREATE TABLE Categorie (
  IdCat INTEGER PRIMARY KEY,
  NomCat VARCHAR(50)
);
```

Table Produit :

```
CREATE TABLE Produit (
  Nump INTEGER PRIMARY KEY,
  Nom VARCHAR(50),
  Prix REAL,
  IdCat INTEGER,
  FOREIGN KEY (IdCat) REFERENCES Categorie(IdCat)
);
```

Un produit ne peut appartenir qu'à une catégorie existante.

3.4 Contrainte conditionnelle — CHECK

Elle impose une **règle logique**.

```
Prix REAL CHECK (Prix > 0)
```

Autre exemple :

```
Age INTEGER CHECK (Age >= 18)
```

3.5 Valeur par défaut — DEFAULT

```
Stock INTEGER DEFAULT 0
```

Si on n'indique rien, le stock vaut 0 automatiquement.

3.6 Nommer les contraintes — CONSTRAINT

```
CONSTRAINT pk_commande PRIMARY KEY (NumC)
```

Utile pour la maintenance et la suppression plus tard.

Exemple COMPLET de création de tables (cas réel)

Table Client

```
CREATE TABLE Client (
  IdClient INTEGER PRIMARY KEY,
  Nom VARCHAR(50) NOT NULL,
  Prenom VARCHAR(50),
  Email VARCHAR(100) UNIQUE
);
```

Table Commande

```
CREATE TABLE Commande (
  NumC INTEGER PRIMARY KEY,
  DateCommande DATE,
  IdClient INTEGER,
```

```
    FOREIGN KEY (IdClient) REFERENCES Client(IdClient)
);
```

4) Suppression d'une table — DROP TABLE

```
DROP TABLE Produit;
```

Supprime :

- la structure
- les données
- les contraintes
- les relations

Bref… plus rien.

5) Modification d'une table — ALTER TABLE

5.1 Ajouter une colonne

```
ALTER TABLE Produit
ADD Stock INTEGER;
```

5.2 Ajouter une colonne avec contrainte

```
ALTER TABLE Produit
ADD Seuil INTEGER CHECK (Seuil > 0);
```

5.3 Supprimer une colonne

```
ALTER TABLE Produit
DROP COLUMN Seuil;
```

5.4 Ajouter une contrainte après création

```
ALTER TABLE Produit
ADD CONSTRAINT prix_min CHECK (Prix > 0);
```

5.5 Supprimer une contrainte

```
ALTER TABLE Produit
DROP CONSTRAINT prix_min;
```

6) Les vues en SQL — CREATE VIEW

6.1 Définition

Une **vue** est :

- Une **table virtuelle**
- Basée sur une **requête**
- Qui ne stocke pas les données
- Mais qui les **affiche dynamiquement**

6.2 Pourquoi utiliser une vue ?

- Sécurité
- Simplification des requêtes
- Masquage de certaines colonnes
- Optimisation
- Tableaux de bord décisionnels

6.3 Syntaxe générale

```
CREATE VIEW nom_vue (colonnes)
AS requête
WITH CHECK OPTION;
```

6.4 Exemples concrets de vues

Exemple 1 : Vue des produits chers

```
CREATE VIEW Produits_Chers (Nump, Nom, Prix)
AS
SELECT Nump, Nom, Prix
FROM Produit
WHERE Prix > 100
WITH CHECK OPTION;
```

Impossible d'insérer dans cette vue un produit à 80 €.

Exemple 2 : Vue des commandes avec noms clients

```
CREATE VIEW Commandes_Client
AS
SELECT Commande.NumC, Client.Nom, Client.Prenom,
Commande.DateCommande
FROM Commande
JOIN Client ON Commande.IdClient = Client.IdClient;
```

Vue très utilisée dans les logiciels de gestion.

Exemple 3 : Vue de stock faible

```
CREATE VIEW Stock_Faible
AS
SELECT Nom, Stock, Seuil
FROM Produit
WHERE Stock < Seuil;
```

Vue parfaite pour la gestion d'alertes.

6.5 Suppression d'une vue

```
DROP VIEW Produits_Chers;
```