



## Thème 3 : Base de données

Leçon 2 : Langage SQL et Gestion de Données

## 1. Introduction : Du Modèle au Réel

Dans le chapitre précédent, nous avons étudié le modèle relationnel. C'était une vision théorique et mathématique : des relations (tables), des attributs (colonnes) et des domaines.

Mais comment passer de cette théorie à un fichier utilisable sur un ordinateur ? C'est le rôle du SGBD (Système de Gestion de Base de Données).

### 1.1. Qu'est-ce qu'un SGBD ?

Un SGBD (ou *DBMS* en anglais) est le logiciel qui sert d'intermédiaire entre l'utilisateur (ou un programme Python, un site web) et les données stockées physiquement sur le disque dur.

- Il garantit l'intégrité : Il vérifie que vous ne mettez pas du texte dans un champ date.
- Il gère la sécurité : Il gère les accès et les mots de passe.
- Il gère la concurrence : Il permet à plusieurs personnes de lire/écrire en même temps sans conflit.

Exemples de SGBD connus : PostgreSQL, MySQL, Oracle Database, Microsoft SQL Server.

### 1.2. Le Langage SQL

Pour "parler" au SGBD, on utilise un langage universel : le SQL (Structured Query Language).

Ce n'est pas un langage de programmation comme Python (il n'y a pas de boucles for classiques ou de fonctions au sens algorithmique), c'est un langage déclaratif.

- En Python (impératif), on dit à la machine *comment faire*.
- En SQL (déclaratif), on dit à la machine *ce que l'on veut obtenir*, et le SGBD se débrouille pour le trouver.

## 2. Notre Base de Données : "Ciné-NSI"

Pour illustrer ce cours, nous allons gérer une petite base de données de cinéma. Voici le schéma relationnel :

1. Realisateur (id\_real, nom, prenom, nation)
2. Film (id\_film, titre, annee, #id\_real)
3. Seance (id\_seance, date\_heure, salle, #id\_film)

Analyse du schéma :

- La table Realisateur contient les informations sur les personnes.
- La table Film contient les films. L'attribut id\_real est une clé étrangère qui fait référence à id\_real de la table *Realisateur*. Cela permet de dire "Ce film a été réalisé par cette personne".
- La table Seance permet de savoir quand un film est projeté. id\_film est une clé étrangère vers la table *Film*.

### 3. Interroger les données

La commande reine du SQL est SELECT. Elle permet de faire une projection (choisir les colonnes) et une sélection (choisir les lignes).

#### 3.1. La requête de base

Pour tout récupérer dans la table des films :

```
SELECT * FROM Film;
```

- \* (étoile, wildcard ) signifie "toutes les colonnes".
- FROM indique la table source.

Pour ne récupérer que le titre et l'année (C'est une projection) :

```
SELECT titre, annee FROM Film;
```

#### 3.2. Filtrer les lignes avec WHERE

C'est ici que l'on applique la logique booléenne. On ne veut garder que les lignes qui respectent une condition (Sélection).

Exemple : Trouver les films sortis après 2010.

```
SELECT titre, annee
FROM Film
WHERE annee > 2010;
```

Les opérateurs logiques :

En NSI, vous connaissez and, or, not. En SQL, c'est pareil : AND, OR, NOT.

Exemple : Les films de science-fiction (imaginons un attribut genre) sortis entre 2000 et 2010.

```
SELECT titre
FROM Film
WHERE annee >= 2000 AND annee <= 2010;
```

#### 3.3. La recherche floue (LIKE)

Parfois, on ne connaît pas le titre exact. L'opérateur LIKE utilise le symbole % comme joker (n'importe quelle suite de caractères).

Exemple : Tous les films dont le titre contient "Star".

```
SELECT * FROM Film WHERE titre LIKE '%Star%';
```

#### 3.4. Trier et dédoublonner

- ORDER BY : Pour trier le résultat.
  - ASC (par défaut) : Croissant.
  - DESC : Décroissant.
- DISTINCT : Pour éviter d'afficher deux fois la même ligne identique.

Exemple : Afficher les années de sortie des films, du plus récent au plus ancien, sans répétition.

```
SELECT DISTINCT annee
FROM Film
ORDER BY annee DESC;
```

### 3.5. Les fonctions d'agrégation

Ce sont des fonctions statistiques qui écrasent les lignes pour donner un résultat unique.

- COUNT(\*) : Compte le nombre de lignes.
- MAX(attribut) / MIN(attribut) : Maximum / Minimum.
- AVG(attribut) : Moyenne.

Exemple : Combien avons-nous de films dans la base ?

```
SELECT COUNT(*) AS nb_total_films FROM Film;
```

*Note : AS permet de donner un alias (un surnom) à la colonne de résultat pour que ce soit plus lisible.*

## 4. Les Jointures : Croiser les tables

C'est le point le plus important des bases relationnelles. Les données sont éclatées dans plusieurs tables pour éviter la redondance. Pour reconstruire l'information complète, il faut les "joindre".

Imaginez que vous avez le film "Inception". Dans la table Film, vous avez seulement id\_real = 4. Qui est 4 ?

Il faut aller voir dans la table Realisateur.

La commande est JOIN ... ON ....

Syntaxe générale :

```
SELECT T1.colonne, T2.colonne
FROM Table1 AS T1
JOIN Table2 AS T2 ON T1.cle_etrangere = T2.cle_primaire;
```

Exemple concret : Afficher le titre du film et le nom de son réalisateur.

```
SELECT Film.titre, Realisateur.nom, Realisateur.prenom
FROM Film
JOIN Realisateur ON Film.id_real = Realisateur.id_real;
```

💡 Explication détaillée pour l'élève :

1. Le SGBD prend la première ligne de Film.
2. Il regarde son id\_real.
3. Il cherche dans Realisateur la ligne qui a le même id\_real.
4. Il "colle" les deux lignes ensemble pour créer une super-ligne temporaire.
5. Il passe à la ligne suivante.

## 5. Insérer et mettre à jour les données

Attention, ces commandes modifient l'état de la base de données de manière permanente.

### 5.1. Insérer (INSERT INTO)

On ajoute un nouvel enregistrement.

```
INSERT INTO Realisateur (nom, prenom, nation)
VALUES ('Nolan', 'Christopher', 'UK');
```

*Attention : Si id\_real est en auto-incrémentation, on ne le précise pas, le SGBD s'en charge. Sinon, il faut le donner.*

### 5.2. Mettre à jour (UPDATE)

On modifie une valeur existante.

**Danger** 💀 : Si vous oubliez le WHERE, toutes les lignes de la table seront modifiées !

Exemple : On corrige l'année du film "Avatar" (disons id 12).

```
UPDATE Film
SET annee = 2009
WHERE id_film = 12;
```

### 5.3. Supprimer (DELETE)

On supprime une ou plusieurs lignes.

**Danger** 💀 : Comme pour UPDATE, sans WHERE, vous videz la table.

```
DELETE FROM Seance
WHERE date_heure < '2023-01-01';
```

## 6. Les Contraintes d'Intégrité

Le SGBD veille au grain. Il existe deux contraintes majeures à comprendre pour le Bac NSI :

1. Contrainte de Domaine : Vous ne pouvez pas mettre "Toto" dans une colonne définie comme INTEGER. Le type doit être respecté.
2. Contrainte de Référence (Clé étrangère) : C'est la plus subtile.
  - Si vous essayez de supprimer le réalisateur "Christopher Nolan" dans la table Realisateur, mais qu'il existe encore des films liés à lui dans la table Film... le SGBD va bloquer la suppression.
  - Pourquoi ? Car sinon, les films se retrouveraient "orphelins" (ils pointeraient vers un id\_real qui n'existe plus), ce qui rendrait la base incohérente.

## 7. Exercice d'application

Soit la requête suivante sur notre base Ciné-NSI :

```
SELECT F.titre, S.date_heure  
FROM Seance AS S  
JOIN Film AS F ON S.id_film = F.id_film  
WHERE F.titre LIKE 'Harry Potter%'  
ORDER BY S.date_heure ASC;
```

Question : Expliquez avec vos propres mots ce que fait cette requête.

Réponse attendue :

Cette requête récupère le titre du film et l'heure de la séance. Elle effectue une jointure entre les séances et les films pour faire le lien. Elle ne garde que les films dont le titre commence par "Harry Potter". Enfin, elle affiche les résultats chronologiquement (du plus vieux au plus récent).