langage SQL

Nous avons vu les modèles de conception de bases de données. L'objectif de ce chapitre est d'expliquer comment interroger et manipuler une base de donnée pour par exemple mettre à jour des données et comment sélectionner des données à partir de critères divers. Pour ce faire, on va utiliser le langage SQL (pour Structured Query Language).

Pour nos exemples on va se baser sur le MCD suivant :


MCD

1. Le langage SQL

SQL est un langage de programmation permettant de gérer des bases de données relationnelles. Un système de gestion de base de données (SGBD) est un logiciel qui permet de stocker, manipuler et interroger des bases de données. Les SGBD les plus connus sont MySQL, PostgreSQL, SQLite, Oracle, SQL Server, etc.

> Par convention, tous les mots clés SQL sont écrits en majuscules.

Une instruction termine par un point-virgule `;`.

2. Construire des requêtes d'interrogation

Une requête SQL est une instruction qui permet d'éxécuter une action sur une base de données. Il existe de nombreux types de requêtes qui sont définit par des mots clés SQL.

2.1. le mot clé SELECT

Pour choisir quels attributs on veut récupérer, on utilise le mot clé `SELECT` avec le nom des attributs séparées par des virgules. Pour sélectionner tous les attributs, on utilise l'astérisque `*` à la place des noms d'attributs (mais c'est déconseillé car cela peut ralentir la requête).

Pour choisir depuis quelle table on veut récupérer les données, on utilise le mot clé `FROM` suivi du nom de la table.

La syntaxe sera donc :

```
SELECT attribut1, attribut2, ...
FROM nom_table;
```

Si on veut récupérer les titres et les années de sortie de tous les livres, on peut :

```

```

> Pour récupérer une seule fois chaque valeur, on peut rajouter le mot clé `DISTINCT` après `SELECT`.

> Par exemple, si on veut récupérer tous les auteurs, on écrira : `SELECT DISTINCT nom FROM auteur;`

2.2 le mot clé WHERE

Pour filtrer les données, on utilise le mot clé `WHERE` suivie d'une condition. Les conditions peuvent être des comparaisons, des opérateurs logiques, des tests de présence dans une liste, etc...

La syntaxe du where est :

```
SELECT attribut1, attribut2, ...
FROM nom_table
WHERE condition;
```

Par exemple, si on veut récupérer les titres et les années de sortie des livres publiés entre 2000 et 2010, on écrira :

```
SELECT titre, annee
FROM livre
WHERE annee > 2000 AND annee < 2010;
```

Pour récupérer les informations sur le livre "1984", on peut faire la requête suivante :

```

```

2.3. le mot clé ORDER BY

l'utilisation du mot clé `ORDER BY` permet de trier les résultats selon un attribut précis. Pour trier dans l'ordre décroissant, on ajoute le mot clé `DESC` après le nom de l'attribut. Par défaut, le tri se fait dans l'ordre croissant ce qui est équivalent à `ASC`.

La syntaxe sera donc :

```
SELECT attribut1, attribut2, ...
FROM nom_table
ORDER BY attribut [ASC|DESC];
```

Pour récupérer tous les livres triés par année de sortie, la requête sera :

```
SELECT titre, annee
FROM livre
```

```
ORDER BY annee ASC;
```

Pour récupérer tous les auteurs triés par ordre alphabétique, on écrira :

2.4. Les jointures

On peut récupérer des données depuis plusieurs tables. Essayez de récupérer les titres de tous les livres et de tous les auteurs. Quel problème rencontrez-vous ?

Pour résoudre ce problème, on utilise les jointures. Il existe plusieurs types de jointures, mais nous allons nous concentrer sur un seul type : la jointure interne.

Pour faire une jointure on utilise le mot clé `JOIN` suivie du nom de la table à joindre et de la condition de jointure.

Dans le MCD, la table `livre` est reliée à la table `auteur` par clé étrangère `auteur`.

Pour écrire cette jointure, on va faire :

```
SELECT titre, prenom, nom
FROM livre
JOIN auteur ON livre.auteur = auteur.id;
```

Le mot clé `ON` est suivi de la condition de jointure. Dans notre cas, on joint les tables `livre` et `auteur` sur l'attribut `auteur` de la table `livre` et l'attribut `id` de la table `auteur`.

Essayez maintenant de récupérer les titres de tous les livres avec le nom de leur auteur.

3. Construire des requêtes de modification

3.1 Ajouter des données

Pour ajouter des enregistrement dans une table, on utilise `INSERT INTO` suivi du nom de la table et des valeurs à insérer. La syntaxe est la suivante : `INSERT INTO nom_table (attribut1, attribut2, ...) VALUES (valeur1, valeur2, ...), (valeurA, valeurB, ...), ...;`

> Il n'est pas nécessaire de spécifier tous les attributs. On peut préciser seulement les attributs pour lesquelles on veut insérer des valeurs. L'ordre des attributs doit correspondre à l'ordre des valeurs mais ce n'est pas nécessaire de conserver l'ordre défini dans la table.

Par exemple, pour ajouter un livre, on écrira :

```

```

A quoi doit-on faire attention lorsqu'on ajoute un livre dans la base de données ?

```

```

3.2 Modifier des données

On peut modifier des données avec le mot clé `UPDATE`. On lui passe le nom de la table, l'attribut à modifier et la nouvelle valeur.

`UPDATE nom_table SET attribut = valeur WHERE condition;` L'instruction va modifier toutes les lignes qui satisfont la condition.

Si on ne met pas de condition, que va-t-il se passer ?

```

```

Comment modifier l'année de sortie du livre "1984" pour la mettre à 1984 ?

```

```

3.3 Supprimer des données

Pour supprimer des données, on utilise le mot clé `DELETE FROM` suivi du nom de la table et de la condition de suppression.

Pour supprimer tous les livres publiés avant 2000 :

```

```

4. Fonctions utiles

Il existe des fonctions pour agréger les données. Par exemple, on peut compter le nombre de résultats, calculer la somme, la moyenne, le minimum, le maximum, etc.

On peut utiliser ces fonctions avec le mot clé `SELECT` pour calculer des statistiques sur les données.

Voici quelques fonctions utiles :

- `COUNT` : pour compter le nombre de résultats
- `SUM` : pour calculer la somme
- `AVG` : pour calculer la moyenne
- `MIN` : pour trouver la valeur minimale
- `MAX` : pour trouver la valeur maximale

Par exemple, si on veut afficher total le nombre de livres, on écrira :

```sql
SELECT COUNT(*)
FROM livre;
```

Si on veut calculer la moyenne des années de sortie des livres, on écrira :

```

```