Requêtes de sélection SQL

Bases de données

1 Présentation du SQL

SQL (*Structured Query Language*, langage structuré de requêtes) est le langage standardisé pour interagir avec les bases de données relationnelles. Il a été créé chez IBM dans les années 1970 et est aujourd'hui utilisé par tous les SGBD majeurs (SQLite, PostgreSQL, MySQL, Oracle, etc.).

SQL se décompose en deux grandes parties :

— le **LDD** (langage de définition des données) : `CREATE TABLE`, `ALTER TABLE`, `DROP TABLE` ;

— le **LMD** (langage de manipulation des données) : `SELECT`, `INSERT`, `UPDATE`, `DELETE`.

Ce chapitre se concentre sur les requêtes de **sélection** (`SELECT`), qui permettent de consulter les données sans les modifier.

2 Table d'exemple

Pour illustrer les requêtes, on utilisera la table **Produits** :

id	nom	categorie	prix	stock
1	Cahier A4	Papeterie	2.50	150
2	Stylo bleu	Papeterie	1.20	300
3	Clé USB 32 Go	Informatique	8.90	45
4	Cartouche encre	Informatique	25.00	20
5	Classeur	Papeterie	3.80	80
6	Souris sans fil	Informatique	15.50	60
7	Colle	Papeterie	1.50	200
8	Casque audio	Informatique	35.00	15

3 SELECT et FROM

La requête la plus simple affiche toutes les colonnes de toutes les lignes d'une table :

```sql
SELECT *
FROM Produits;
```

Le symbole $*$ signifie « tous les attributs ». Pour n'afficher que certaines colonnes, on les nomme explicitement :

```sql
SELECT nom, prix
FROM Produits;
```

④ WHERE : FILTRER LES RÉSULTATS

La clause `WHERE` permet de ne garder que les lignes qui vérifient une condition.

```sql
SELECT nom, prix
FROM Produits
WHERE categorie = 'Informatique';
```

4.1 Opérateurs de comparaison

Opérateur	Signification
`=`	Égal
`<>` ou `!=`	Différent
`<`, `>`	Inférieur, supérieur strict
`<=`, `>=`	Inférieur ou égal, supérieur ou égal
`BETWEEN a AND b`	Compris entre a et b (inclus)
`LIKE`	Comparaison avec motif (`%` = joker)

4.2 Combiner des conditions : AND, OR, NOT

```sql
SELECT nom, prix
FROM Produits
WHERE categorie = 'Informatique' AND prix < 20;
```

Résultat : Clé USB 32 Go (8.90), Souris sans fil (15.50).

⑤ ORDER BY : TRIER LES RÉSULTATS

```sql
SELECT nom, prix
FROM Produits
WHERE prix < 5
ORDER BY prix;
```

Par défaut, le tri est **croissant** (`ASC`). Pour trier par ordre décroissant, on ajoute `DESC` :

```sql
SELECT nom, prix
FROM Produits
ORDER BY prix DESC;
```

6 DISTINCT : éliminer les doublons

Le mot-clé `DISTINCT` élimine les lignes identiques dans le résultat.

```sql
SELECT DISTINCT categorie
FROM Produits;
```

Résultat : Papeterie, Informatique.

7 Fonctions d'agrégation

Les fonctions d'agrégation calculent une valeur à partir d'un ensemble de lignes.

Fonction	Rôle
`COUNT(*)`	Nombre de lignes
`SUM(attribut)`	Somme des valeurs
`AVG(attribut)`	Moyenne des valeurs
`MIN(attribut)`	Valeur minimale
`MAX(attribut)`	Valeur maximale

Combien de produits ont un stock inférieur à 50 ?

```sql
SELECT COUNT(*)
FROM Produits
WHERE stock < 50;
```

Résultat : 3.

Quel est le produit le plus cher ? On utilise une **sous-requête** (notion hors programme, mais commode) :

```sql
SELECT nom, prix
FROM Produits
WHERE prix = (SELECT MAX(prix) FROM Produits);
```

Résultat : Casque audio (35.00).

8 Complément : regrouper les lignes avec GROUP BY

Complément hors programme, utilisé dans les TP. La clause `GROUP BY` regroupe les lignes qui ont la même valeur pour un attribut ; la fonction d'agrégation est alors calculée pour chaque groupe. Le mot-clé `AS` donne un nom à une colonne du résultat.

Combien de produits compte chaque catégorie ?

```sql
SELECT categorie, COUNT(*) AS nb_produits
FROM Produits
GROUP BY categorie;
```

Résultat : Informatique 4, Papeterie 4.

On peut trier les groupes avec `ORDER BY`, puis ne garder que les premières lignes du résultat avec `LIMIT` :

```sql
SELECT categorie, SUM(stock) AS stock_total
FROM Produits
GROUP BY categorie
ORDER BY stock_total DESC
LIMIT 1;
```

Résultat : Papeterie (730), la catégorie qui a le plus d'articles en stock.

⑨ Synthèse

— `SELECT … FROM` affiche des colonnes d'une table ; `*` désigne toutes les colonnes ;

— `WHERE` filtre les lignes selon une condition ;

— `ORDER BY` trie les résultats (`ASC` par défaut, `DESC` pour l'ordre décroissant) ;

— `DISTINCT` élimine les doublons dans le résultat ;

— les fonctions d'agrégation (`COUNT`, `SUM`, `AVG`, `MIN`, `MAX`) calculent des valeurs synthétiques ;

— l'ordre des clauses est toujours : `SELECT` → `FROM` → `WHERE` → `ORDER BY`.