

# Les requêtes SQL



Le but de ce cours est de décrire la structure d'une requête SQL afin de sélectionner et mettre à jour des données. La sélection va consister en l'écriture de requêtes SQL permettant de trouver toutes les données de la base vérifiant un certain critère.

Le premier rôle du programmeur de bases de données va donc être de traduire des questions que l'on se pose sur les données du langage naturel au langage SQL, afin que le SGBD puisse y répondre.

## 1. Sélection de données

### Sélection de colonnes avec la clause **SELECT**

La commande **SELECT** permet de sélectionner des colonnes d'une ou plusieurs tables données en paramètre. Comme souvent, l'ensemble des colonnes peut être indiqué par une étoile \*. La syntaxe est :

```
SELECT colonne(s) FROM nom_table(s)
```

Exemple :

Prenons pour exemple la table TABLE\_NOTES dont le schéma relationnel et les enregistrements sont les suivants :

nom	maths	anglais	info
Joe	19	17	15
Alan	14	15	17
Zoe	18	16	20

Les requêtes SQL à base d'instruction **SELECT** suivantes renvoient les tables :

```
SELECT * FROM TABLE_NOTES ;
```

→ Renvoie →

nom	maths	anglais	info
Joe	19	17	15
Alan	14	15	17
Zoe	18	16	20

```
SELECT maths, nom FROM TABLE_NOTES ;
```

→ Renvoie →

maths	nom
19	Joe
14	Alan
18	Zoe

### Sélection de lignes avec la clause **WHERE**

La commande **WHERE** filtre des enregistrements à partir d'un critère de sélection. Cette instruction vient en complément de l'instruction **SELECT**. La syntaxe est :

```
SELECT colonne(s) FROM nom_table(s) WHERE critère_de_sélection
```

## Exemples :

```
SELECT * FROM TABLES_NOTES  
WHERE maths>15 OR info>18 ;
```

→ Renvoie →

nom	maths	anglais	info
Joe	19	17	15
Zoe	18	16	20
Zoe	18	16	20

```
SELECT maths, nom FROM TABLES_NOTES  
WHERE info>15 ;
```

→ Renvoie →

maths	nom
19	Zoe
18	Alan

## Suppression des données redondantes

Le premier exemple du tableau précédent fait apparaître des valeurs de retour redondantes (nom : Zoe). La commande **DISTINCT** associée à **SELECT** permet de supprimer ces doublons.

```
SELECT DISTINCT nom FROM TABLES_NOTES  
WHERE maths>15 OR info>18 ;
```

→ Renvoie →

nom
Joe
Zoe

## Tri de donnée avec ORDER BY

Le tri par ordre alphabétique des données de retour sur une colonne de référence peut être effectué avec l'instruction **ORDER BY**. Le mot clé **ASC** stipule un tri croissant et **DESC** un tri décroissant.

```
SELECT * FROM TABLES_NOTES  
ORDER BY anglais ASC ;
```

→ Renvoie →

nom	maths	anglais	info
Alan	14	15	17
Zoe	18	16	20
Joe	19	17	15

## Les fonctions de groupes

Les fonctions de groupes permettent d'obtenir des informations sur un ensemble de lignes travaillant sur les colonnes et non pas sur les lignes comme avec **WHERE**.

- x **AVG** : calcule la moyenne d'une colonnes
- x **SUM** : calcule la somme d'une colonnes
- x **MIN, MAX** : calculent le minimum et le maximum d'une colonnes
- x **COUNT** : donne le nombre de lignes d'une colonnes

```
SELECT AVG(maths), MIN(maths),  
MAX(maths) FROM TABLES_NOTES ;
```

→ Renvoie →

17	14	19
----	----	----

## Renommage avec la clause AS

Les attributs des données renvoyées peuvent être renommées avec **AS** :

```
SELECT nom AS name FROM  
TABLES_NOTES ;
```

→ Renvoie →

name
Alan
Zoe
Joe



## Les jointures avec JOIN

Une jointure permet d'associer dans une même requête plusieurs tables en les liant à partir d'un attribut commun. Par exemple, supposons que nous ayons une table contenant des mentions en fonctions de la note obtenue, sur le principe suivant :

mention	note
Bien	14
Presque parfait	19
Inespéré	18

*Extrait de la table Table\_mentions*

Associer directement une mention à chaque élève en fonction de son résultat en mathématiques s'écrit :

```
SELECT Table_notes.nom,  
Table_mentions.mention FROM Table_notes JOIN  
Table_mentions ON  
Table_notes.maths = Table_mentions.note ;
```

→ Renvoie →

nom	mention
Alan	Bien
Zoe	Inespéré
Joe	Presque parfait

## 2. Modification d'une table

### Mise à jour d'une table avec INSERT INTO

La mise à jour d'une table avec un nouvel enregistrement s'effectue en entrant le tuple de nouvelles valeur dans la table avec la commande **INSERT INTO** :

```
INSERT INTO table (attr1, attr2, ...) VALUES (val1, val2 ...)
```

Par exemple, l'ajout d'une nouvelle appréciation s'écrit :

```
INSERT INTO Table_mentions (note, mention) VALUES (20 , 'Au top') ;
```

### Modification d'un enregistrement avec UPDATE

La modification de valeurs d'un enregistrement existant s'effectue avec la commande **UPDATE** :

```
UPDATE table SET (attr1 = val1, attr2 = val2...)
```

Par exemple la requête suivante change la note de mathématiques de Joe avec 15 :

```
UPDATE Table_notes SET maths=15 WHERE Nom='Joe' ;
```

### Supprimer un enregistrement

La suppression d'une ligne s'effectue avec la commande **DELETE** :

```
DELETE FROM table WHERE condition
```

La requête suivante supprime tous les enregistrements qui ont une note de maths  $\leq 14$



```
DELETE FROM Table_notes WHERE maths <= 14 ;
```

## Types de données en SQL

Les domaines abstraits du modèle relationnel correspondent à des types de données en langage SQL. Le tableau suivant récapitule les principaux types utilisés :

Nom du type	Description
<b>Numériques</b>	
<b>SALLINT</b>	Entier 16 bits signé
<b>INTEGER</b>	Entier 32 bits signé
<b>BIGINT</b>	Entier 64 bits signé
<b>Decimal(t, f)</b>	Décimal signé de t chiffres dont f après la virgule
<b>REAL</b>	Flottant 32 bits (valeur approchée)
<b>DOUBLE PRECISION</b>	Flottant 64 bits (valeur approchée)
<b>Chaîne de caractères</b>	
<b>CHAR(n)</b>	Chaîne de n caractères (espaces dans caractères manquants)
<b>VARCHAR(n)</b>	Chaîne d'au plus n caractères
<b>TEXT</b>	Chaîne de taille quelconque
<b>Date</b>	
<b>DATE</b>	Date au format 'AAAA-MM-JJ'
<b>TIME</b>	Heure au format 'hh:mm:ss'
<b>DATETIME</b>	Instant au format 'AAAA-MM-JJ hh:mm:ss'

