Requêtes SQL

L'ordre "SELECT"

L'ordre SELECT est la commande de base du SQL. Elle sert à extraire des données d'une base ou à calculer de nouvelles données à partir de données existantes.

Syntaxe générale d'une commande 'SELECT'

```
SELETE [DISTINCT|ALL] {* | liste_de_colonnes }
FROM nom_de_table
[WHERE predicat]
[GROUP BY liste_des_colonnes_du_groupage]
[HAVING predicat]
[ORDER BUY liste_des_colonnes_de_tri]
```

La clause SELECT est obligatoire et recense les colonnes que l'on veut voir figurer dans le résultat de la requête.

La clause FROM est obligatoire et permet de préciser la table sur laquelle s'applique l'ordre SELECT.

La clause WHERE est optionnelle et permet de préciser des conditions sur les données. Elle agit comme un filtre. C'est une expression logique (prédicat) qui ne peut aboutir qu'à un calcul booléen et donc un résultat VRAI, FAUX ou NULL (absence d'information).

Les clauses optionnelles GROUP BY et HAVING s'utilisent en général conjointement et permettent d'ajouter des conditions, non plus sur les données présentes dans les tables en jeu, mais par rapport à des résultats calculés.

La clause ORDER BY, optionnelle, permet de trier le résultat de la requête dans l'ordre des colonnes précisées.

Sélection des noms et prénoms des joueurs :

```
mysql>select Nom, prenom from Joueurs;
```

L'opérateur *

Le caractère * (étoile) placé dans la clause SELECT récupère toutes les colonnes de la table précisée dans la clause FROM de la requête.

Utilisation de l'opérateur *

```
mysql> select * from Joueurs;
```

Note :L'utilisation systématique de l'opérateur * dans la clause SELECT n'est pas conseillée. En effet, cet opérateur oblige le SGBDR à retrouver par lui-même le nom explicite des colonnes et se révèle donc un peu plus coûteux en ressources que la spécification directe du nom des colonnes.

L'opérateur ''DISTINCT'' (ou ''ALL'')

Lorsque le moteur du SGBDR construit une réponse à une requête, il rapatrie toutes les lignes correspondantes, généralement dans l'ordre où il les trouve, même si ces dernières sont en double (`ALL` par défaut). C'est pourquoi il est parfois nécessaire d'utiliser le mot-clé `DISTINCT` qui élimine les doublons dans la réponse.

Exemple:

Recherche des postes des joueurs (ALL par défaut)

```
mysql> select poste from Joueurs;
```

Recherche des postes des joueurs de l'OL avec le mot-clé DISTINCT

```
mysql> select distinct poste from Joueurs;
```

L'opérateur ''AS''

L'opérateur `AS` permet de renommer des colonnes ou/et d'en rajouter.

Exemple d'utilisation de AS

```
mysql> select Nom, date_de_naissance AS  'Ne le'
         from Joueurs;
```

La clause ''WHERE''

Le prédicat de la clause `WHERE` doit contenir n'importe quelle expression logique renvoyant une valeur logique. Seules les lignes de la table répondant `Vrai` à la condition seront retenues.

On peut y utiliser les opérateurs logiques classiques :

Opérateurs	**Symboles**
Comparaisons	= <> < <= > >=
Logiques	OR AND
Négation	NOT
Parenthèses	( )

Recherche des joueurs dont le prénom est Gregory

```
mysql> select Prenom, Nom from joueurs where prenom='Gregory';
```

```
Recherche de Gregory Coupet
mysql> select Prenom, Nom from Joueurs where Prenom='Gregory' and Nom='Coupet';
```

Recherche des joueurs dont le numero est inférieur à 11

```
mysql>select * from Joueurs where Numero <= 11;
```

La clause "ORDER BY"

La clause `ORDER BY` permet de visualiser les réponses dans un ordre donné.

Syntaxe

```
ORDER BY colonne1 { [ ASC ] | DESC } [, colonne2 { [ ASC ] |DESC } ] ...
```

ou Syntaxe

```
ORDER BY 1 { [ASC] | DESC} [, 2 { [ ASC] | DESC} ] ...
```

Les chiffres 1, 2, … sont relatifs à l'ordre des colonnes exprimées dans la clause `SELECT`.

Recherche des noms de joueurs comme précedemment, mais trié par leur Nom

```
mysql> select * from Joueurs order by Nom;
```

```
Par défaut, l'ordre de la clause ORDER BY est l'ordre ascendant (ASC). Pour obtenir un tri décroissant, il faut spécifier le mot-clé DESC pour chacune des colonnes que l'on veut trier.
```

Il est possible de spécifier plusieurs clés de tri: une clé primaire suivi d'une clé secondaire, etc.

Les colonnes clé servant de tri peuvent être référencées dans la clause `ORDER BY` par leur numéro d'apparition dans la clause `SELECT`

Traitement des chaînes de caractères

Il existe plusieurs opérateurs et fonctions permettant de travailler sur les chaînes de caractères.

Concaténation (CONCAT)

La fonction `CONCAT` permet de concaténer (ajouter bout à bout) des champs de type de caractères

Utilisation de la fonction CONCAT

```
mysql> select concat (Nom, ', ', Prenom) from Joueurs;
```

Recherche partielle (LIKE)

L'opérateur `LIKE` permet d'effectuer une comparaison partielle. Il est employé pour les colonnes contenant des donnees de type chaîne de caractères et utilise les caractères % et _ pour désigner n'importe quelle chaîne de caractères (y compris la chaîne vide) ou un seul caractère respectivement.

Recherche des prénoms commençant par G

```
mysql> select concat (Nom, ', ', Prenom), Poste
  from Joueurs where Prenom like 'G%';
```

Distinction entre majuscules et minuscules (LOWER, UPPER)

Les fonctions `LOWER` et `UPPER` permettent de mettre en majuscules ou en minuscules des chaînes de caractères dans les requêtes.

Utilisation de LOWER ou UPPER

```
mysql> select upper(Nom), Prenom from Joueurs where Poste like 'Avant %';
```

On peut aussi grâce à ces fonctions effectuer des recherches sans tenir compte de la casse:

recherche dans tenir compte de la casse

Remplacement (REPLACE)

La fonction REPLACE permet de remplacer une chaîne de caractère par une autre.

Syntax: REPLACE (chaîne, source, cible)

Exemple :

```
mysql> select  Nom, replace (Poste, 'Gardien', 'Goal')
    from joueurs where Poste='Gardien';
```

Traitement numérique

SQL intégre les fonctions élémentaires du calcul numérique +, -, * et /. Mais il existe aussi bien d'autres fonctions.

opérations élémentaires

```
mysql> select  3+5, 2-5, 3/4, 2*3;
```

Les fonctions mathématiques (MySQL):

Syntaxe	**Fonction**
ABS(n)	Valeur absolue
ACOS(n)	Cosinus réciproque
ASIN(n)	Sinus réciproque
ATAN(n)	Tangente réciproque
CEILING(n)	Borne supérieure entière de n
COS(n)	Cosinus
EXP(n)	Exponentiel
FLOOR(n)	Borne inférieure entière de n
LN(n)	Logarithme népérien
LOG(n)	Logarithme décimal
MOD(n, m)	Modulo (reste de la division entière de n par m
PI()	Valeur de Pi
POWER(n, m)	n à la puissance m
RAND()	Nombre aléatoire entre 0 et 1
ROUND(n)	Entier le plus proche de n
ROUND(n,m)	Nombre le plus proche de n, à m chiffres après la virgule près
SIGN(n)	Renvoie -1, 0 ou 1 suivant que le nombre est négatif, nul ou positive
SIN(n)	Sinus
SQRT(n)	Racine carrée
TAN(n)	Tangente

Fonctions de traitement des données temporelles

En SQL, il est possible de définir et manipuler des données temporelles.

Deux sortes de données sont accessibles: la date (jour, mois, année) et le temps (heures, minutes, secondes), séparément ou ensemble.

Utilisation de la date et du temps courant

```
mysql> select current_date , current_time,current_timestamp;
```

La particularité de SQL est d'afficher la date sous la forme d'une chaîne de caractères 'AAAA-MM-JJ' (Année, Jour, Mois). Le temps est donné sous la forme 'HH:MM:SS'.

Sélection des joueurs nés après le 1er janvier 1980

```
mysql> select Nom, Date_de_naissance from Joueurs where Date_de_naissance > '1980-01-01';
```

Il est possible d'extraire une partie d'une date ou d'un temps grâce à la fonction `EXTRACT`.

Syntaxe: `EXTRACT( { YEAR | MONTH | DAY | HOUR | MINUTE | SECOND } FROM …`

Fonctions d'agrégation

Il est possible de réaliser aussi des statistiques sur des colonnes à l'aide de différents opérateurs:

Opérateur	**Signification**
AVG	Moyenne
MAX	Maximum
MIN	Minimum
SUM	Total
COUNT	Nombre

Pour les fonctions `COUNT`, `SUM`, ou `AVG` il est possible d'effectuer la statistique en évitant les doublons éventuels. Cela est possible grâce au mot-clé `DISTINCT`.

La clause "GROUP BY"

La clause `GROUP BY` permet d'effectuer un regroupement d'enregistrements.

Nombre de joueurs par poste

```
mysql> select Poste, count (Poste) from Joueurs group by Poste;
```

Opérateur IN

L'opérateur `IN` permet de chercher si une valeur se trouve dans un ensemble donné: une liste.

Syntaxe: valeur `IN` (valeur1, [valeur2 …])

Recherche des joueurs prénommés Grégory ou Rémy

```
mysql> select Nom, Prenom from Joueurs where Prenom in ('Gregory', 'Remy');
```

Opérateur BETWEEN

L'opérateur `BETWEEN` permet de rechercher si une valeur se situe dans un intervalle donné (*valeurs incluses*).

Syntaxe: valeur `BETWEEN` valeur1 `AND` valeur2

La clause "HAVING"

La clause `HAVING` s'exécute après que les clauses `SELECT` et `WHERE` ont donné des résultats. Elle crée un filtre supplémentaire sur les résultats et non sur les données.

rechercher les postes peu fournies

```
mysql> select  Poste, count(Poste) from Joueurs group by Poste
       having count(Poste) <= 2;
```

Requêtes sur plusieurs tables

Il est possible d'effectuer en SQL des requêtes sur plusieurs tables à la fois.

Le surnom

Il est possible de donner un surnom à une table dans la clause `FROM`.

Syntaxe: `FROM` table surnom

Dès lors, ce surnom peut être utilisé pour préciser de quelle table provient une colonne. Cela s'effectue à l'aide de la notation pointée.

utilisation des surnoms

```
mysql> select J.Numero, J.Nom from Joueurs J;
```

Les clés

Clé primaire (PRIMARY KEY)

Une *clé primaire* (ou *index primaire* ou *identifiant*) permet de retrouver sans ambiguïté la ligne d'une table. Elle est donc unique au sein de la table.

Clé étrangère (FOREIGN KEY)

Une clé étrangère est une clé d'une table insérée dans une autre table afin d'assurer une relation entre les deux tables.

Dans la table `Joueurs`, la clé `JoueurID` est unique par joueur. La clé `ClubID` est une clé

étrangère dans la table `Joueurs` et une clé primaire dans la table `Club`.

Contenu de la table Club

Les jointures

Les jointures permettent de relier les tables entre elles.

Sans elles, l'interrogation de différentes tables simultanément conduit au produit cartésien des enregistrements.

Interrogation sans jointure

```
mysql> select C.ClubID, C.Nom, J.Numero, J.Nom, J.ClubID
       fromClub C, Joueurs J
       where J.Numero > 36;
```

Jointure simple

La jointure s'effectue en imposant une condition d'égalité entre des valeurs de deux colonnes d'une table.

Un exemple de jointure

```
mysql> select C.ClubID, C.Nom, J.Numero, J.Nom, J.ClubID  from Club C, Joueurs J
where J.Numero > 36 and C.ClubID=J.ClubID;
```

Jointure interne (INNER JOIN)

Il est possible d'écrire de manière équivalente la jointure précédente directement dans la partie `FROM` en utilisant `INNER JOIN`:

Utilisation de INNER JOIN ... ON ...

```
mysql> select C.ClubID, C.Nom, J.Numero, J.Nom, J.ClubID  from Club C inner join
Joueurs J on C.ClubID=J.ClubID where J.Numero > 38 ;
```

Dans le cas où les clés sont identiques entre les deux tables:

Utilisation de INNER JOIN ... USING ..

```
mysql> select C.ClubID, C.Nom, J.Numero, J.Nom, J.ClubID  from Club C inner join
Joueurs J using (ClubID) where J.Numero > 38 ;
```

Jointure externe (LEFT/RIGHT JOIN)

Une jointure externe permet de garder les enregistrements qui n'ont pas de correspondance avec la table sur laquelle on effectue une jointure.

Utilisation de LEFT JOIN

```
mysql> SELECT C.ClubID, C.Nom, J.Numero, J.Nom, J.ClubID FROM Joueurs J LEFT
JOIN Club C ON J.ClubID=C.ClubID;
```

Utilisation de RIGHT JOIN

```
mysql> SELECT C.ClubID, C.Nom, J.Numero, J.Nom, J.ClubID FROM Joueurs J RIGHT
JOIN Club C ON J.ClubID=C.ClubID;
```

Comptage des joueurs par club avec LEFT JOIN

```
mysql> SELECT C.Nom AS Club, COUNT(J.Nom) AS "NOMBRE DE JOUEURS"
    -> FROM Joueurs J LEFT JOIN Club C USING (ClubID)
    -> GROUP BY C.ClubID ;
```

Comptage des joueurs par club avec RIGHT JOIN

```
mysql> SELECT C.Nom AS Club, COUNT(J.Nom) AS "NOMBRE DE JOUEURS"
    -> FROM Joueurs J RIGHT JOIN Club C USING (ClubID)
    -> GROUP BY C.ClubID ;
```

Les sous-requêtes

SQL permet d'imbriquer des ordres `SELECT` au sein de requêtes de type `SELECT`. On parle alors de *sous-requêtes*.

En fonction des résultats d'une sous-requête SQL qui fournit :

- soit plusieurs colonnes et plusieurs lignes
- soit plusieurs colonnes et une seule ligne
- soit une seule colonne sur plusieurs lignes
- soit une seule colonne et une seule ligne
- soit aucun résultat

Il est possible d'écrire une requête à l'aide de sous-requêtes dans les clauses SELECT ou WHERE.

Un exemple de clauses SELECT imbriquées

```
mysql> select Numero, Nom
       from Joueurs where Numero > 25 and
       ClubID = (select ClubID from Club where Nom="Olympique Lyonnais");
```