Commencer rapidement le SQL

iThorgrim

Tutoriel par @Nobodie

Prélude : Ce sujet n'est pas original, il a été transféré d'un précédent forum d'émulation en date d'Octobre 2013. J'en suis l'auteur(Nobodie), je le transfère donc pour les besoins de ce forum. Bien qu'il soit en partie ré-écrit, il se peut que certains termes soient inadaptés ou ne soient plus d'actualité.

Ce tutoriel n'a pas été directement écrit par moi-même, mais par un des membres de l'équipe Mist-Eria. ce sujet est posté avec l'accord de son auteur. Des modifications peuvent y être apportées dans un but de clarification.

Salut à tous,

Si vous avez ouvert ce sujet, c'est probablement en vous demandant pourquoi un nouveau tutoriel sur le SQL serait utile. J'aimerais pouvoir dire que celui que j'écris est révolutionnaire (c'est à la mode comme mot :D), mais ce n'est pas le cas. Pourtant si je le rédige c'est bien dans le simple but d'offrir un autre point de vu d'apprentissage de ce langage ; ici le but ne sera pas d'avoir l'explication la plus fine du fonctionnement du SQL, mais d'avoir une connaissance pratique et efficace au plus vite (notamment dans le cadre d'un émulateur WoW). Par conséquent, ce tutoriel est supra-condensé et nécessitera peut-être de multiples relectures de votre part. Certains diront peut-être que c'est anti-pédagogique, moi je dirai que c'est une autre pédagogie :P .

Sans plus attendre voici le sommaire :

On a du pain sur la planche, donc on y va.

1) Qu'est-ce que le SQL ?

Première question et pas des moindre : qu'allons-nous étudier ? SQL est l'acronyme de Structured Query Language ou langage de requête structurée. Il permet de dialoguer avec des bases de données de différents types. A partir de ce constat, vient une deuxième question : Qu'est-ce qu'une base de données ? Une base de données (ou DB pour database) est une sorte de conteneur informatique pour organiser et stocker de nombreuses données.

Pour gérer ces bases de données, on utilise des logiciels dit SGDB (Système de Gestion de Base de Données). Il en existe de très nombreux types, du système peu connu utilisé par votre libraire pour gérer ses stocks jusqu'au système ultra perfectionné utilisé par l'armée. En ce qui nous concerne, nous utiliserons un système utilisant pour dialoguer le SQL (sinon le cours perd tout son intérêt). Ils sont aussi très nombreux : Postgre, MySQLi, MariaDB, Microsoft SQL Serveur, MySQL et bien d'autres. Un des plus répandus est MySQL, c'est celui-ci que nous utiliserons durant ce tutoriel ; sachez cependant que tous les systèmes utilisant le SQL sont proches à quelques petites différences près.

MySQL a un fonctionnement dit serveur, cela signifie qu'il fonctionne en couple avec d'un côté un serveur qui exécute et donne les informations demandées par le biais de requêtes à un client.

Après ce premier chapitre, vous devriez avoir compris que le SQL est un langage utilisé pour dialoguer avec un serveur gérant une base de données. Votre objectif est donc simple, comprendre ce langage et l'assimiler.

2) Installation d'un environnement local et présentation de PhpMyAdmin

2) Installation d'un environnement local et présentation de PhpMyAdmin

Si vous avez suivi la partie précédente, vous avez compris qu'il vous faudra un logiciel qui fera serveur et un autre qui vous permettra de communiquer avec le serveur (le client). Pour travailler chez vous, il vous faudra un environnement local, c'est à dire entre autre de quoi travailler avec un logiciel serveur sur votre propre pc. Lorsque vous travaillerez en production, vous n'aurez plus votre serveur sur votre PC, mais même si le serveur est ailleurs, le logiciel client restera le même.

Pour avoir un environnement local, il existe différents logiciels, mais ceux qui sont graphiques et qui ne font que MySQL sont à ma connaissance inexistants, par conséquent, nous devons utiliser des solutions un peu plus complètes qui proposent d'autres logiciels serveurs ; sous Windows, mon choix s'est porté pour UwAmp qui est très léger et très clair, et propose en plus du logiciel serveur l'application cliente. Pour mac, il existe mamp et sous linux on peut installer lamp ; ou bien installer MySQL-server seul puis dans les deux cas installer PhpMyAdmin (qui sera le logiciel client) (ou tout autre logiciel client).

Je vous laisse installer ces trois logiciels (et un serveur MySQL de préférence (lien)) seuls selon votre système et essayer de les prendre en main. J'entends par "les prendre en main" savoir lancer le serveur, ce qui correspond souvent à un bouton de type Lancer / Start / Démarrer ou à une commande sous linux (je vous laisse faire vos petites recherches).

C'est bon ? Vous avez installé votre serveur et vous l'avez lancé ? Et bien maintenant, il va falloir lancer le logiciel client. Il en existe beaucoup, mais dans notre cas, nous en utiliserons un assez particulier car il se lance via le navigateur : PhpMyAdmin. Pour le faire fonctionner deux choix s'offrent à vous :

- Soit votre logiciel vous propose directement d'ouvrir la page par l'intermédiaire d'un bouton (c'est le cas avec UwAmp)
- Soit vous lancer votre navigateur, vous vous rendez sur **http://localhost/phpmyadmin** ou **http://localhost:8888/phpmyadmin**

Si vous n'êtes pas connecté automatiquement, les identifiants sont probablement : Pseudo root et Mot de passe : rien, root ou un mot de passe que vous aurez choisi durant l'installation.

- Dans le menu de droite, on voit la liste des bases de données qui sont disponibles sur mon serveur local
- Le menu en haut vous permet d'accéder à la majorité des fonctionnalités du client
- La partie centrale vous donne quelques informations générales sur votre serveur

Je pourrai vous expliquer chaque page et quel est leur intérêt, mais vous verrez bien vite leur but au fur et à mesure de ce "cours". De plus, vous pouvez très bien les visiter par vous-même (en plus elles sont très claires ^^).

On passe donc à la suite

3) Mise en place d'une base de données

Si vous souhaitez exécuter votre première requête, il vous faudra d'abord une base de données avec au moins une table, c'est pourquoi c'est ce que nous allons commencer à faire.

Pour cela, sélectionnez l'onglet Base de données et donnez un nom à votre futur base de données. Pour l'exemple, j'ai choisie de prendre le cas d'une librairie qui aurait besoin d'un système pour connaitre ses stocks de livre.

Une fois que vous avez fait cette action, dans le menu de droite votre base de données sera apparue.

Il ne vous reste plus qu'à cliquer dessus et on vous proposera de créer une nouvelle table, dans notre cas, ce sera livre et la table comportera cinq champs.

On vous demande maintenant tout plein d'informations pour faire votre table, pas d'inquiétude, cette page bien qu'effrayante au premier abord a pour but de vous aider. On vous demande principalement deux informations : le nom de chaque colonne et son type, c'est à dire le type de donnée que contiendra la colonne. Cela peut être un nombre entier (INT), décimal (FLOAT), un petit champ de texte (VARCHAR), un grand champ de texte (TEXT), une date, etc ...

Tout à l'heure, j'ai dit que mon exemple comprendrait 5 colonnes, les voici :

- titre qui sera un VARCHAR
- auteur qui sera aussi un VARCHAR
- prix qui sera un FLOAT
- description qui sera un TEXT

- id qui sera un INT assez particulier : il sera l'index primaire (les lignes s'organiseront par ordre croissant selon l'id) et il aura une valeur auto-incrémentée, cela signifie que vous n'aurez pas à lui donner de valeur seul, MySQL lui attribuera tout seul un entier qui comme valeur l'id précédent + Le but est de pouvoir facilement identifier vos champs, les sélectionner mais surtout qui permet d'optimiser la vitesse de recherche.

Vous remarquerez que généralement on attribue la valeur 255 aux varchars, c'est en faites le nombre de caractères que pourra contenir le champ (pour des raisons plus ou moins complexe).

Nous allons maintenant un peu remplir cette table, cliquez donc sur le nom de votre table puis dans le menu sur Insérer (vous verrez plus tard comment insérer des lignes en écrivant vous-même le code SQL). Remplissez les champs comme vous le souhaitez en respectant le type de données (pas de lettre dans un INT ou un FLOAT). N'oubliez pas que nous avons défini le champ id en auto-incrémentation, par conséquent vous ne devez pas le remplir, MySQL s'en occupe pour vous. Réitérez cette insertion 2 ou 3 fois pour remplir un peu la table.

En cliquant sur l'onglet Affiché, vous devriez maintenant voir vos lignes.

Félicitations, votre première table est déjà quelque peu remplie. Nous allons pouvoir commencer à entrer dans le vif du sujet :) .

4) Votre première requête et les critères de recherches

Sur une base de données, on peut vouloir faire de nombreuses actions via le SQL, la première et la plus simple est le fait de sélectionner des informations venant d'une table.

Pour envoyer une requête SQL au serveur MySQL, il vous faut cliquer sur l'onglet SQL et rédiger votre requête.
Nous avons donc décidé que nous souhaitions sélectionner des informations. Remettons nous dans le cas du libraire, il veut voir tous les livres dont il dispose. Voici la requête qu'il devra taper (nous la décortiquerons juste après) :

```
SELECT * FROM livres;
```

Essayez cette requête, vous devriez obtenir un résultat semblable à celui de la page Afficher. Et oui, PhpMyAdmin pour faire fonctionner ses pages utilise lui-même des requêtes SQL. Si vous retourner sur la page Afficher, vous verrez même que la requête que vous avez taper est noté en haut de la page.

Voici comment il faut lire la requête (bien que le serveur ne la lise pas vraiment dans cette ordre pour des questions d'exécutions plus optimisés) :

- SELECT -> sélectionne
- * -> toutes les colonnes
- FROM -> depuis
- livres -> la table que vous souhaitez lire (ici livres)
- ; -> le point-virgule ou délimiteur désigne la fin d'une requête, bien que PHPMyAdmin fonctionne bien sans qu'on le précise, il faut toujours le mettre, c'est une habitude à prendre qui vous permettra de ne jamais avoir de petit problème. C'est notamment grâce au ; que MySQL sait où commence la requête suivante dans le cas où vous en exécutez plusieurs.

Ce qui donne plus clairement : Sélectionne toutes les colonnes depuis la table livres.

Maintenant que vous avez exécuté votre première requête (Félicitations au passage), nous allons essayer d'approfondir l'utilisation de cette commande. Effectivement, vous avez ici la base même d'une requête SELECT, mais elle peut être très facilement développé.

La première chose que vous remarquez dans mon commentaire de la requête, c'est que l'étoile désigne toutes les colonnes vous avez ; donc vous pouvez devinez qu'on peut ne choisir que certains champs : si vous souhaitez sélectionner uniquement titre et le prix du livre vous pouvez modifier la requête de tout à l'heure ce qui vous donnera :

```
SELECT titre, prix FROM livres;
```

Après exécution seule les colonnes demandées sont affichés.

Vous n'y avez peut-être pas encore pensé mais lorsque vous faites un "FROM livres", le serveur MySQL qui lit la requête ne devine pas tout seul où se trouve la table livres (dans quelle DB ?). Mais si vous vous rappelez à peu près du début, vous aurez remarqué qu'avant de cliquer

sur SQL, nous avons clique sur notre base de données, il a donc conclus que c'était dans la base de données librairie. Mais si par heureux hasard, vous souhaitiez faire une requête sur une table qui n'est pas dans la db "où vous vous trouvez", il y a une solution : plutôt que d'écrire le nom de la table normalement, on va le précéder du nom de la DB de cette manière :

```
SELECT * FROM librairie.livres;
```

Maintenant que vous commencez à comprendre cette requête, nous allons étudier les critères de sélection. Effectivement, bien que la requête SELECT de base puissent paraître pratique, lorsque vous aurez 36 000 lignes dans votre table, il ne sera pas forcément aisé de la retrouvé ^^. C'est pour cela qu'ont été créés les critères de sélection, nous allons donc en voir quelques-uns.

Le premier des critères de sélection est le critère WHERE : il vous permet entre autres de comparer une colonne à une valeur et de ne retourner que ceux qui correspondent. Vous souhaitez connaitre tous les livres qui coûtent moins de 10 euros ? Il y a une requête pour cela :

```
SELECT titre, prix FROM livres WHERE  prix < 10;
```

Vous pouvez aussi utiliser le signe égal. Dans le cas ou vous souhaiteriez comparer un texte ou une chaîne de caractères plus courte, vous devez utiliser des guillemets de cette façon :

```
SELECT * FROM livres WHERE auteur="J.R.R. Tolkien";
```

Et vous obtiendrez uniquement la ligne correspondant à Tolkien.

Avec ce critère de sélection vous pouvez déjà bien vous amuser, mais il y en a deux autres qui sont plutôt important : ORDER BY et LIMIT. A chaque fois que vous faites une recherche, vos résultats sont classés selon l'ID (car c'est la clé primaire). Cependant vous pouvez avoir envie de changer cela sur un de vos résultats pour une raison quelconque : c'est le rôle d'ORDER BY : classez selon l'ordre que vous voulez, comme tous les critères de sélection, il se place à la fin de la requête :

```
SELECT * FROM livres ORDER BY prix;
```

La requête est plutôt explicite, toutes les lignes sont classés par ordre croissant selon le prix, si vous souhaitez avoir le résultat par ordre décroissant, il vous suffit d'ajouter DESC à la fin du ORDER BY :

```
SELECT * FROM livres ORDER BY prix DESC;
```

Enfin, il reste le LIMIT. Jusqu'à maintenant (et même plus tard), nous n'avons travaillé que sur de toute petite base mais lorsque vous travaillerez sur des bases de milliers de ligne, une recherche basique devra vous renvoyer des milliers de ligne ce qui n'est pas forcément très rapide. Pour cela, on a inventé un autre critère qui permet de limiter le nombre de résultat, par défaut PhpMyAdmin vous met la limite pour que vous n'ayez pas trop de résultats, mais il est toujours bon de savoir comment cela fonctionne : Il existe deux types de syntaxes LIMIT : [...] LIMIT x et [...] LIMIT y, x. Si vous ne mettez qu'un seul chiffre, vous aurez les x premiers résultats, si vous en mettez deux, vous aurez les résultats entre x et y : s'il y a 50 résultats et que vous faites : LIMIT 25, 50 vous aurez les 25 derniers résultats. Etant donné la taille de la table sur laquelle nous travaillons, nous ne pouvons pas vraiment testé ce critère, mais ne vous inquiétez pas, cela viendra bien vite ^^.

Comme vous l'aurez compris, les critères de sélection vous permettront de faire bien des choses, mais si vous les combinez, les possibilités deviennent faramineuses ^^. Votre libraire peut très bien faire une recherche demandant :

Tous les livres à moins de 10 euros classé par prix croissant dans la limite de 10 résultats :

```
SELECT * FROM livres WHERE prix < 10 ORDER BY prix LIMIT 10;
```

Vous pouvez aussi faire deux WHERE : tous les livres coutant plus de 10 euros et ayant pour auteur George Martin :

```
SELECT * FROM livres WHERE prix > 10 AND auteur="George Martin";
```

5) Les autres requêtes du CRUD

CRUD est l'acronyme anglais pour Create, Read, Update, Delete. Il désigne les quatre opérations de base pour la gestion des données, et ça tombe bien, parce qu'est exactement ce dont ont à besoin pour faire du SQL :). Après tout, actuellement vous avez beau pouvoir

tombé bien, parce que c'est exactement ce dont ont à besoin pour faire du SQL :) . Après tout, actuellement vous avez beau pouvoir sélectionner tout ce que vous souhaitez dans une base de données, ça ne vous empêche pas de ne pas pouvoir vous amuser ^^.

Le CRUD correspond donc à 4 requêtes différentes. Nous avons déjà eu l'occasion de faire le Read avec le SELECT, il n'en reste plus que 3.

L'insertion de données

Précédemment quand vous souhaitiez créer une nouvelle ligne, vous passiez par l'onglet Insertion, cela a beau être pratique, ce n'est pas disponible sur tout les logiciels clients et surtout, ce n'est pas très optimisés lorsqu'on a une longue liste à insérer.

Pour insérer une ligne dans une table, on utilise la requête REPLACE INTO. Voici ça syntaxe :

```
REPLACE INTO table (`nom du champ 1`, `nom du champ 2`) VALUES (`valeur 1`, `valeur 2`);
```

Si vous souhaitez faire plusieurs insertions d'un coup, il suffit de rajouter une virgule à la fin et de remettre de nouvelles valeurs de cette manière :

```
REPLACE INTO table (`nom du champ 1`, `nom du champ 2`) VALUES
(`valeur 1`, `valeur 2`),
(`valeur 1`, `valeur 2`),
(`valeur 1`, `valeur 2`);
```

C'est toujours pratique ^^.

N'oubliez pas que la colonne id (et de manière générale les colonnes en auto-incrémente) ne doit pas être remplit sauf dans certains cas particuliers.

C'est à peu près tout ce qu'il y a à savoir sur cette requête.

La suppression

La suppression correspond à la requête : DELETE. Attention, cette requête est très dangereuse une mauvaise utilisation de cette dernière peut vider complètement une table !

C'est sûrement la requête avec la syntaxe la plus simple :

```
DELETE FROM livres;
```

N'essayez pas cette requête ou vous videriez toute votre table pour rien. Et oui, cette requête a toujours besoin d'un critère de recherche pour ne pas faire trop de dégâts. Dans ce genre de cas, l'ID est très pratique, il suffit de faire un WHERE id=x pour supprimer exactement la ligne que l'on souhaite, mais vous pouvez aussi supprimer selon d'autres critères. Si notre libraire trouve que ce n'est pas rentable de vendre des livres à moins de trois euros, il peut les supprimer de sa DB de cette façon :

```
DELETE FROM livres WHERE prix < 3;
```

Cette requête est indispensable et pourtant très dangereuse, vérifiez toujours deux fois avant de lancer ce type de requêtes.

La mise à jour

Eh oui, nous voici déjà à la dernière requête élémentaire : la mise à jour d'une ligne. Tout comme le DELETE FROM, cette requête est relativement dangereuse : si vous ne mettez pas de critères de recherche, tous les champs seront mis à jour ce qui n'est peut-être pas très pratique et peu causer beaucoup d'ennuie (ça sent le vécu :P).

La requête correspondant à la mise à jour est : UPDATE, sa syntaxe est :

```
UPDATE table SET champ = "nouvelle valeur" AND champ2 = "nouvelle valeur 2";
```

Dans le cas de notre libraire qui veut vendre plus chère un livre, il devra faire :

```
UPDATE livre SET prix = 15 WHERE id=1;
```

Vous pouvez aussi vouloir augmenter une valeur sans pour autant savoir quelle était la valeur de base. Il y a une solution à ce problème : plutôt que de faire SET valeur = "nouvelle valeur", vous pouvez faire : SET valeur = valeur + x.

Si notre libraire veut augmenter de 1 euro "Le Silmarillion", il peut exécuter :

```
UPDATE livres SET prix = prix +1 WHERE titre = "Le Silmarillion";
```

S'il veut, il peut aussi tenter d'augmenter le prix de tous ses livres de 1 :

```
UPDATE livres SET prix = prix +1;
```

Ce type d'UPDATE est très dangereux tout comme le DELETE, quoique plus facile à annuler (il suffit de refaire la requête avec un SET prix = prix-1). Toutefois sur des requêtes très complexes, il est très difficile de l'annuler.

Vous disposez maintenant de base en SQL et vous pouvez d'ores et déjà commencer à vous amuser, mais sachez que vous ne connaissez qu'une infime partie de ce qui est faisable avec ce langage ;) .

6) Jointures et autres réjouissances

A venir avec au programme :

- Quelques agrégats bien utiles
- Les jointures internes
- Les jointures externes
- Les sous-requêtes
- Quelques liens vers des tutoriels sur les SP et les requêtes préparés

Si vous avez d'autres idées, n'hésitez pas à me les soumettre sur ce topic :) .

7) Mémo

Lorsqu'on débute on peut rapidement avoir l'impression qu'on ne se rappellera pas des nombreuses syntaxes. Pas d'inquiétude à avoir, ce n'est pas très étonnant, pour vous aider à les mémoriser, voici un récapitulatif des différentes syntaxes vues :

La sélection des données

```
SELECT * FROM table;
SELECT colonne1, colonne2 FROM db.table;
```

Les critères de sélections

```
[...] WHERE champ = "valeur";
[...] WHERE champ > "valeur";
[...] WHERE champ = "valeur" AND champ2 < "valeur2";
[...] ORDER BY id;
[...] ORDER BY prix DESC;
[...] LIMIT 10;
[...] LIMIT 10, 20;
[...]  WHERE champ = "valeur" ORDER BY id DESC LIMIT 60;
```

Les autres requêtes du CRUD

```
REPLACE INTO table (champ1, champ2) VALUES ("valeur 1", "valeur 2");
REPLACE INTO table (champ1, champ2) VALUES ("valeur 1", "valeur 2"), ("valeur 1", "valeur 2");
DELETE FROM table WHERE champ = "valeur";
UPDATE table SET champ="valeur" WHERE champx="valeurx";
```

J'ai bien conscience que ce tutoriel est très condensé, n'hésitez donc pas à le relire. Deux ou trois passages ont été légèrement simplifiés, mais l'idée générale reste là et ça ne devrait pas vous poser de problème ^^.

Sur ceux, bonne continuation, et bon développement.

Thiaz

J'ai une petite question, pourquoi tu utilises "REPLACE INTO" au lieu de "INSERT INTO" ? Je trouve ça "maladroit" de présenter ça a des débutants sans au moins présenter les différences. Surtout que "INSERT INTO" est quand même plus parlant. En théorie les deux commandes insert bien des nouvelles données, mais si tu veux éviter de remplacer une valeur déjà existante (qui a une PRIMARY KEY et/ou un index UNIQUE) il vaut mieux utiliser INSERT INTO et avoir une erreur et être sûr de ce que tu fais (pour des débutants encore une fois).

Mais merci pour le tuto, je suis sûr qu'il aidera beaucoup de monde :D
