MySQL en lignes de commande

Vous n'êtes pas obligé de passer par '*phpMyAdmin*' pour gérer votre base de données. Vous pouvez le faire en lignes de commande, ce qui vous donne un contrôle complet sur toutes les procédures.

A cette fin, vous devez ouvrir une invite de commandes et vous placer dans le répertoire bin de MySQL. Par exemple : `cd c:\mamp\bin\mysql\bin`

Dans le répertoire, saisir la commande suivante : `mysql –P 8081 –h localhost -u root –p`

Le port (-P) et l'host (-h) ne sont pas obligatoires si vous utilisez le localhost avec son port par défaut. MySQL vous invite alors à saisir votre mot de passe : `enter password :`

Cette formalité remplie, la connexion à MySQL est établie et le client attend vos commandes.
`mysql>`

Vous pouvez gérer vos procédures en lignes de commande et retenez bien que toutes les commandes dans MySQL se termine par **;**.

Pour quitter le client, tapez la commande suivante : `exit`

Création d'une base de données

La création d'une base se fait de la façon suivante (ne pas oublier l'encodage) :
```
CREATE DATABASE nom_base CHARACTER SET 'utf8';
```

La création de la base se fait à partir du compte ROOT. Mais pour des raisons de sécurité, il est vivement déconseillé d'utiliser ce compte administrateur pour travailler dans une base de données. ROOT gère l'intégrité de MySQL. La moindre erreur de commande pourrait être fatale au server. Donc, dès que la base est créée, même avant, ajoutez-lui son propre utilisateur :
```
GRANT [ALL PRIVILEGES] ON nom_base.* TO 'admin'@'localhost' [IDENTIFIED BY
'mdp'] [WITH GRANT OPTION];
```

GRANT : donne des droits à un utilisateur.
[ALL PRIVILEGES] : tous les droits. Il est possible de ne donner aucun droit avec [USAGE]. Il existe de nombreux autres droits, pour les connaître, référez-vous à la documentation.
ON nom_base.* : base de données sur laquelle l'utilisateur a des droits ou pas. ON *.* est un privilège global, surtout utilisé avec USAGE.
TO 'admin' : nom de l'utilisateur. Les '' ne sont pas obligatoire, mais préférable en cas de caractères spéciaux.
@'localhost' : hôte sur lequel l'utilisateur se connecte. Cela peut être en local, mais aussi à distance, grâce à une adresse IP ou un nom de domaine.
IDENTIFIED BY 'mdp' : défini un mot de passe pour l'utilisateur. Ce n'est pas obligatoire, mais fortement conseillé.
[WITH GRANT OPTION] : donne le droit à l'utilisateur de créer d'autres utilisateurs.

La commande STATUS permet d'obtenir des informations sur la base de données : `STATUS ;`

Il est possible de supprimer une base : `DROP DATABASE [IF NOT EXISTS] nom_base;`

Passez par SHOW DATABASES ; pour lister toutes les bases existantes.

Pour quitter mysql : `QUIT ;`

A partir de ce moment, il est possible de se connecter directement sur la base en lançant MySQL :
`mysql –h localhost –u admin –p nom_base`

Une fois le mot de passe saisi, vous êtes directement connecté à la bonne base avec les bons droits.

Exemple

```
mysql> CREATE DATABASE microsoftia CHARACTER SET 'utf8';
mysql> GRANT ALL PRIVILEGES ON microsoftia.* TO admin@localhost IDENTIFIED BY 'mdp'
WITH GRANT OPTION;
mysql> quit;
C:\Server\MySQL\bin>mysql -h localhost -u admin -p microsoftia
```

A tout moment, il est possible de changer de base, à partir du moment que vous êtes connecté avec les bons privilèges, grâce à la commande : `USE nom_base;`

Création de tables

L'élaboration des tables est l'élément central d'une base de données. Des tables mal conçues rendent la base difficile à exploiter. Avant même de créer la table, il est important de réfléchir à ses éléments. Bien choisir les colonnes qui composent la table, puis les typer selon les besoins. Connaître les valeurs limites et savoir si la colonne peut être nul ou non. Enfin, il faut réfléchir à la clé primaire.

La clé primaire est l'identifiant de l'enregistrement. Si une colonne permet d'identifier un et un seul enregistrement, s'il est unique, alors c'est la clé primaire. Sinon, il faut penser à ajouter une colonne pour ça. En général c'est un entier, positif et non nul et auto-incrémenté. Cette clé primaire permet de respecter la contrainte d'unicité.

L'ajout d'une table se fait par la commande CREATE TABLE, comme dans l'exemple ci-dessous :

```
CREATE TABLE [IF NOT EXISTS] nom_table(
       colonne_1 SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
       colonne_2 VARCHAR(40) DEFAULT 'default',
       colonne_3 CHAR(1),
       colonne_4 DATETIME NOT NULL,
       [PRIMARY KEY(colonne_1)]
)ENGINE=INNODB;
```

Les colonnes sont définies par des couples nom/type et éventuellement complétés des clauses.
DEFAULT : force une valeur par défaut. Cette commande ne peut être utilisée qu'avec une constante et pas avec une autre commande.
NOT NULL : interdit qu'une colonne se retrouve sans valeur, c'est à dire à NULL. Par défaut, c'est possible.
AUTO_INCREMENT : la colonne n'a pas besoin d'être remplie, elle est remplie automatiquement par une valeur incrémentée.
PRIMARY KEY(clé) : définit la colonne clé comme clé primaire. Par convention, la clé primaire doit être la première colonne déclarée.
CHECK (condition) : condition pouvant spécifier des plages ou des listes de valeurs possibles.
Les 4 dernières clauses sont ce qu'on appelle des contraintes d'intégrités.

Exemple

```
mysql> CREATE TABLE User(
    -> user_id INT NOT NULL AUTO_INCREMENT,
    -> nom VARCHAR(30),
    -> prenom VARCHAR(20),
    -> annee_naissance YEAR,
    -> PRIMARY KEY(user_id));
mysql> SHOW TABLES;
mysql> DESCRIBE User;
```

Même si idéalement les tables sont parfaitement réfléchies avant leurs créations, il est possible de les modifier grâce à la commande ALTER TABLE.

```
ALTER TABLE Pas_finie
[ADD [COLUMN] colonne_oubliee description_colonne;]
[DROP [COLUMN] nom_colonne ;]
[CHANGE nom_colonne nom_nouveau VARCHAR(10) NOT NULL;]
```

ADD : Ajout d'une nouvelle colonne. Le mot clé est facultatif, MySQL ajoutant, dans ce cas, une colonne par défaut. La description de la nouvelle colonne répond aux mêmes règles qu'à la création.
DROPP : supprime la colonne à partir de son nom.
CHANGE : Change le nom d'une colonne ou sa description. Il est impératif de reprendre la description de la colonne pour que cette dernière ne soit pas modifiée par la commande. Pour simplement modifier la description, il suffit d'indiquer 2 fois le nom de la colonne.

Pour modifier une clé primaire, l'affaire est un peu plus compliquée. Il faut, tout d'abord, enlever l'auto-incrémentation (si elle existe, évidemment). Puis supprimer la clé avant de l'ajouter sur la colonne voulue.

```
-- pas de AUTO_INCREMENT
ALTER TABLE Pas_finie MODIFY colonne_1 INT UNSIGNED NOT NULL;
ALTER TABLE Pas_finie DROP PRIMARY KEY; --supprime la clé primaire
ALTER TABLE Pas_finie ADD PRIMARY KEY(nom_colonne); --nouvelle clé primaire
```

Il est aussi possible de supprimer une table, avec la commande suivante :

```
DROP TABLE nom_table;
```

Enfin, il est utile de pouvoir consulter les tables d'une base :

```
SHOW TABLES; --liste les tables de la bdd
DESCRIBE nom_table; --liste les colonnes de la table demandée.
```

Clé étrangère

La clé étrangère d'une table référence la colonne (ou un groupe de colonnes) d'une autre table. Ainsi, il est impossible d'insérer des données inexistantes dans cette table.

```
CREATE TABLE nom_table(
nom_colonne1 description_colonne1,
[nom_colonne2 description_colonne2,
…]
[[CONSTRAINT [nom_contrainte]] FOREIGN KEY (colonne(s)_clé_étrangère)
REFERENCES table_référence(colonne(s)_clé_référence)]
) [engine=moteur];
```

CONSTRAINT permet de donner un nom à la clé étrangère, même si ce n'est pas obligatoire. Pour bien définir la clé étrangère, il faut absolument que la colonne_clé_étrangère et la même description que la colonne_clé_référence.

Exemple

```
CREATE TABLE User (
    numero INT UNSIGNED PRIMARY KEY AUTO_INCREMENT,
    client INT UNSIGNED NOT NULL,
    produit VARCHAR(40),
    quantite SMALLINT DEFAULT 1,
    CONSTRAINT fk_client_numero
        FOREIGN KEY (client)
        REFERENCES Client(numero)
)ENGINE=InnoDB;    -- MyISAM interdit !
```

Il est évidement possible d'ajouter une clé après la création de la table

```
ALTER TABLE Commande
ADD CONSTRAINT fk_client_numero FOREIGN KEY (client) REFERENCES Client(numero);
```

Et il est possible de l'enlever

```
ALTER TABLE nom_table
DROP FOREIGN KEY symbole_contrainte
```