Création d'une base de données et d'une table

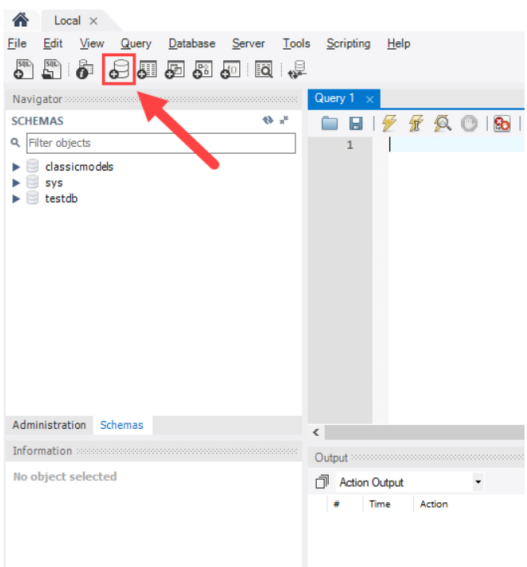
Mise en place

Lancez `MySQL Workbench`.

Cliquez sur votre instance locale dans `MySQL Connections`.

Créer une base de données

Pour créer une nouvelle base de données sur le serveur `MySQL` auquel vous êtes connecté, il suffit de faire `Create new Schema`.



Pour `MySQL` un `schema` est une schéma de base de données. Vous pouvez le voir comme un synonyme de base de données.

Entrez dans `Schema name` par exemple `test`. Ce sera le nom de la base de données.

Cliquez sur `Apply`.

Cela revient exactement à entrer la commande `SQL` suivante :

```
CREATE SCHEMA test;
```

Ou, celle-ci qui est exactement équivalente :

```
CREATE DATABASE test;
```

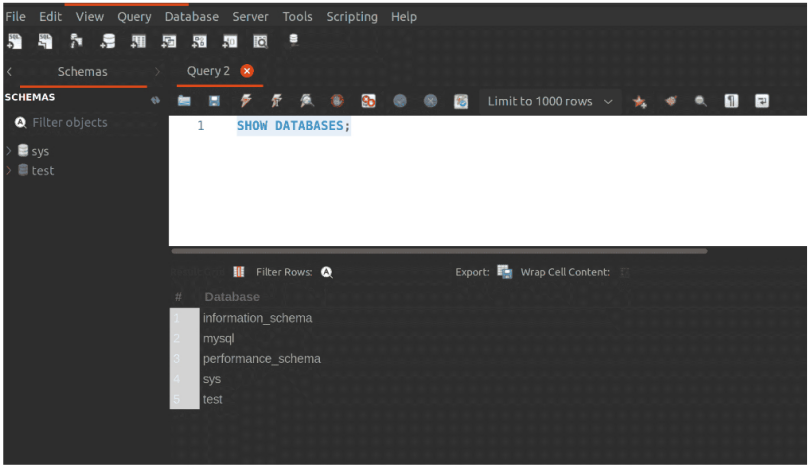
Lister les bases de données

Vous pouvez exécuter cette commande pour lister les bases de données :

```
SHOW DATABASES;
```

Cliquez sur l'éclair pour exécuter la commande.

Vous aurez par exemple :

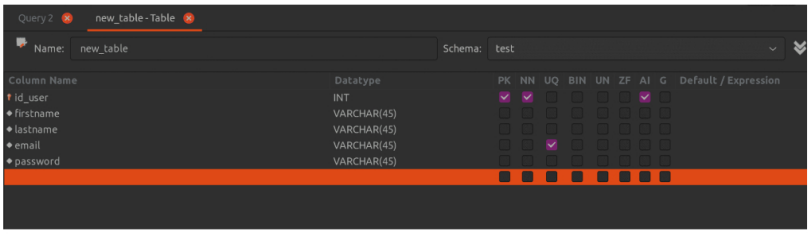


Créer une table

Pour créer une table, il faut utiliser la commande `CREATE TABLE` :

```
CREATE TABLE [IF NOT EXISTS] nom_table(
    colonne1,
    colonne2,
    ...,
    contraintes_table
) ENGINE=moteur;
```

Pour commencer, utilisez le `GUI Workbench` en cliquant sur `Create table` :



Entrez dans `name` : `user`.

Cliquez ensuite sur `Appliquer`.

`Workbench` va alors vous montrer la requête `SQL` qu'il va exécuter :

```
CREATE TABLE `test`.`user` (
  `id_user` INT NOT NULL AUTO_INCREMENT,
  `firstname` VARCHAR(45) NULL,
  `lastname` VARCHAR(45) NULL,
  `email` VARCHAR(45) NULL,
  `password` VARCHAR(45) NULL,
  PRIMARY KEY (`id_user`),
  UNIQUE INDEX `email_UNIQUE` (`email` ASC) VISIBLE);
```

`CREATE TABLE `test`.`user` signifie que dans la base de données `test` nous allons créer une base de données `user`.

Remarquez la syntaxe pour les colonnes : nom, type de données et éventuelles contraintes et attributs.

Par exemple, pour la clé primaire nous avons `INT` comme type de données, la contrainte `NOT NULL` et l'attribut `AUTO_INCREMENT`.

`INT` peut contenir un entier signé ou non signé. Ici, il ne sera pas signé car l'identifiant sera positif. La valeur maximale est `4294967295` ce qui est suffisant.

`AUTO_INCREMENT` permet d'incrémenter automatiquement à chaque insertion l'identifiant de 1 en commençant par la valeur 1.

`VARCHAR(45)` peut contenir une chaîne de caractères de 0 à 45 octets.

`NULL` signifie que la colonne peut contenir des valeurs nulles.

Notez que nous avons la contrainte de table `PRIMARY KEY (`id_user`)` qui permet d'indiquer quelle colonne est la clé primaire.

Notez la deuxième contrainte qui rend la colonne email `UNIQUE`.

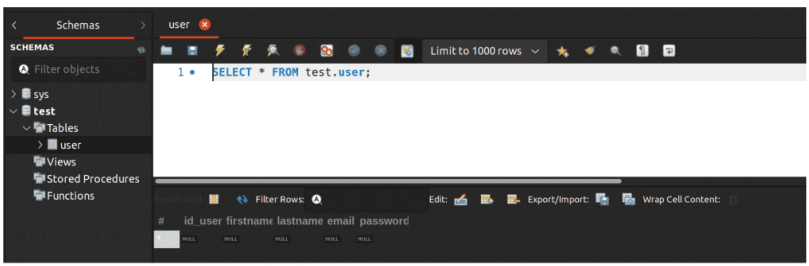
Cliquez ensuite sur `Appliquer`.

Visualiser les données dans une table

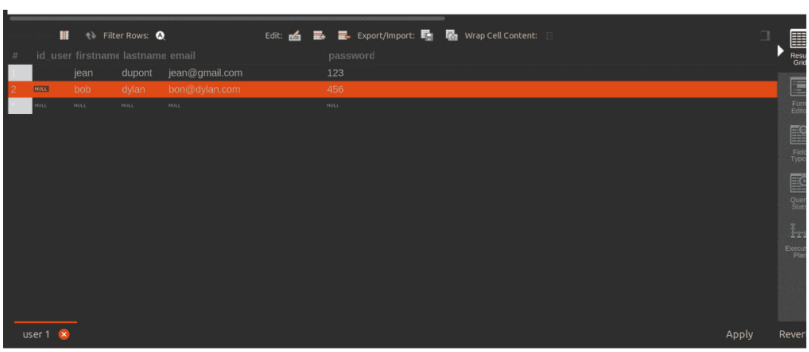
Cliquez sur `test > Tables > user`.

Cliquez du droit sur `user` et sélectionnez `Select rows`.

Cela permet d'afficher toutes les données de la table :



Nous pouvons entrer quelques données dans la table pour nos tests :



Cela exécutera la commande suivante :

```
INSERT INTO `test`.`user` (`id_user`, `firstname`, `lastname`, `email`, `password`)
VALUES ('1', 'jean', 'dupont', 'jean@gmail.com', '123');
INSERT INTO `test`.`user` (`id_user`, `firstname`, `lastname`, `email`, `password`)
VALUES ('2', 'bob', 'dylan', 'bob@dylan.com', '456');
```

`INSERT INTO `test`.`user`` permet de spécifier la table où insérer les données.

Ensuite, nous précisons les colonnes dans l'ordre dans lequel nous passerons les données.

Enfin, nous passons les données après `VALUES` dans l'ordre correspondant à celui des colonnes.

La syntaxe est la suivante :

```
INSERT INTO table(c1,c2,...)
VALUES (v1,v2,...);
```