Gestion des Tables

Maintenant qu'on a notre bdd, il nous reste à créer des tables. Prenons un classique, la table "**user**", une table qui contiendra tous nos utilisateurs.

Une table est un tableau dont chaque ligne est une nouvelle entrée, donc ici, chaque ligne (**row**) de notre table "**user**" sera un nouvel utilisateur.

```
CREATE TABLE users (
    id int,
    username varchar(50),
    email varchar(255),
    password text,
    active bool,
    createdAt datetime
    );
-- le ";" indiquant la fin de notre requête,
-- Cela ne pose aucun problème de sauter des lignes dans une requête.
```

Voyons cette requête en détail :

- "**CREATE TABLE**" est assez explicite. - "**users**" est le nom de ma table.
- Ce que l'on trouve entre les parenthèses sont les colonnes de mon tableau. chaque colonne a un nom suivi de son type, on verra les types principaux ici. Mais vous pouvez trouvez tous les types sur la documentation officiel ou sur : https://www.w3schools.com/mysql/mysql_datatypes.asp

Voyons nos colonnes:

- "**id**" est présente sur une majorité de table, servant de repère et d'index à nos lignes. Ici on lui donne le type "**int**" qui représente un nombre entier.
- "**username**" qui a le type "**varchar(50)**" cela indique un string de 50 caractère max.
- "**email**" qui est de type "**varchar(255)**" 255 est la taille maximum du varchar.
- "**password**" est de type "**text**", un string avec une limite bien plus grande. (utile car les mots de passes hashés peuvent être bien plus grand que leur version de base.)
- "**active**" est de type "**bool**", un boolean, en bdd il sera représenté par "**1**" ou "**0**";
- "**createdAt**" est de type "**datetime**" il prendra donc la date et l'heure.

Cela dit, on a oublié plein de choses dans cette requête SQL. On va donc supprimer cette table et la refaire.

```
DROP TABLE users;
```

Puis nous allons ajouter quelques paramètres :

```
CREATE TABLE users (
    id int NOT NULL AUTO_INCREMENT,
```

```
    username varchar(50) NOT NULL,
    email varchar(255) NOT NULL,
    password text NOT NULL,
    active bool DEFAULT 0,
    createdAt datetime DEFAULT CURRENT_DATE(),
    PRIMARY KEY (id),
    UNIQUE(id),
    UNIQUE(email)
);
```

Que voyons nous de nouveau ici ?

- "**NOT NULL**" apparaît plusieurs fois, il indique que chaque entrée de ce champ doit être rempli, elle ne peut pas être vide.
- "**AUTO_INCREMENT**" qui est sur "**id**" indique que ce champ nombre doit si aucune donnée ne lui est fourni augmenter par lui même de "**1**" à chaque nouvelle entrée.
- "**DEFAULT**" permet de donner une valeur par défaut si jamais rien ne lui est donnée.
- "**CURRENT_DATE()**" est une fonction qui va aller chercher la date acutelle.
- "**PRIMARY KEY(id)**" indique que le champ "**id**" doit être la "**CLEF PRIMAIRE**" de notre table. Cela signifie que c'est ce champ qui servira à indexer nos lignes. C'est celui qui identifiera chacune de nos entrées.
- "**UNIQUE()**" apparaît deux fois et indique que le champ entre parenthèse doit être "**UNIQUE**" C'est à dire qu'une erreur sera retourné si on tente de donner une valeur à ce champ qui existe déjà dans la table.

Je peux vérifier la structure de ma table avec :

```
SHOW CREATE TABLE users;
```

Il m'affichera la requête permettant la création de ma table. si maintenant je fais :

```
SHOW TABLES;
```

Je verrais la liste des tables contenue dans ma BDD actuelle.

Mais zut, il se trouve que je me suis trompé dans ma table !

J'ai mit "**CURRENT_DATE()**" mais c'est un champ "**datetime**" hors je ne vais donc obtenir que la date et non l'heure qui sera par defaut à "**00:00:00**". Je pourrais certe encore une fois tout supprimer puis la reconstruire à zéro, mais ça serait casse pied et si on a déjà des données dans ma table, je perdrais tout.

Heureusement on a une solution :

```
ALTER TABLE users MODIFY COLUMN createdAt datetime DEFAULT CURRENT_TIMESTAMP;
```

- "**ALTER TABLE**" indique que je souhaite modifier une table, puis j'indique laquelle.
- "**MODIFY COLUMN**" indique que je souhaite modifier une des colonnes. J'indique alors laquelle puis je lui donne ses nouvelles caractéristiques.
- "**CURRENT_TIMESTAMP**" me donne le timestamp actuel, duquel je peux tirer l'heure et la date.

Autre changement possible, finalement je veux connaître la date de naissance de mon utilisateur :

```
ALTER TABLE users ADD birthday date NOT NULL;
```

- "**ADD**" après notre "**ALTER TABLE**" permet d'ajouter une nouvelle colonne.

Finalement l'âge ne me convient pas :

```
ALTER TABLE users DROP COLUMN birthday;
```

- "**DROP COLUMN**" nous permet de supprimer une de nos colonnes.