Jointures

Parfois il nous faut faire des requêtes qui sont lié à plusieurs tables à la fois. par exemple j'aimerais avoir le nom des utilisateur et tous les messages avec leur date qui leur sont liés. Je pourrais faire :

```
SELECT id, username FROM users;
```

Puis faire une boucle dans mon langage back et récupérer tous les messages avec :

```
SELECT content, createdAt FROM messages WHERE userId = 1;
-- ... --
SELECT content, createdAt FROM messages WHERE userId = 8;
```

Mais on multiplierait les requêtes et on ferait travailler notre BDD pour rien, c'est là que va intervenir les jointures :

INNER JOIN

```
SELECT users.username, messages.content, messages.createdAt
    FROM users
    INNER JOIN messages
    ON messages.userId = users.id;
```

Quand on fait un join, on a besoin d'indiquer devant chaque colonne, de quel table on parle.
"**users.username**" signifie la colonne "**username**" de la table "**users**".

- "**INNER JOIN**" indique que l'on va joindre une autre table à notre requêtes.
- "**ON**" permet d'indiquer comment on va lier les deux, ici on lie "**messages.userId**" et "**users.id**".

On peut aussi réduire la taille de notre requête grâce aux "**ALIAS**" :

```
SELECT u.username, m.content, m.createdAt
    FROM users u
    INNER JOIN messages m
    ON m.userId = u.id;
```

Ici on a récupéré tous nos utilisateur ayant posté des messages avec leurs messages. Mais que faire si je souhaite récupérer aussi les utilisateur qui n'ont pas de messages ?

LEFT JOIN

```
SELECT u.username, m.content, m.createdAt
    FROM users u
    LEFT JOIN messages m
    ON m.userId = u.id;
```

"**LEFT JOIN**" à la différence de "**INNER JOIN**" va nous retourner tous les résultats de la table de gauche (ICI celle à gauche du "**JOIN**" donc "**users**") et seulement les résultats liés dans celle de droite.

Les colonnes liées aux messages pour les utilisateurs n'en n'ayant pas sont donc "**NULL**".

RIGHT JOIN

```
SELECT u.username, m.content, m.createdAt
    FROM users u
    RIGHT JOIN messages m
    ON m.userId = u.id;
```

"**RIGHT JOIN**" fera l'inverse de "**LEFT JOIN**", nous retournant tous les résultats de la table de droite (ICI celle à droite de "**JOIN**" donc "**messages**"). Dans notre cas, nos messages étant forcément lié à un utilisateur, on ne verra ici pas de différence avec "**INNER JOIN**".

CROSS JOIN

```
SELECT u.username, m.content, m.createdAt
    FROM users u
    CROSS JOIN messages m;
```

"**CROSS JOIN**" est rarement utilisé car il va retourner toute les valeurs des deux tables et les associers même si ils n'ont rien à voir ensemble. De plus il peut produire énormément de résultat puisqu'il tente de faire toute les combinaisons possibles.

SELF JOIN

```
SELECT  u1.username AS Name1,
        u2.username AS Name2
    FROM users u1, users u2
    WHERE u1.id <> u2.id
        AND u1.createdAt = u2.createdAt;
```

Il est possible de joindre une table avec elle même et cela sans mot clef "**JOIN**". Ici on récupère des paires d'utilisateur ayant créé leurs compte au même moment.

EXCLUDING JOIN

Reprenons notre "**LEFT JOIN**" et ajoutons une petite condition :

```
SELECT u.username, m.content, m.createdAt
    FROM users u
    LEFT JOIN messages m
    ON m.userId = u.id
    WHERE m.userId IS NULL;
```

Notre résultat ne nous donne plus que les utilisateurs qui n'ont aucun messages. Cette façon de faire nous permet de récupérer tous ceux qui n'ont pas de jointures seulement.

Exercices

Il existe des extensions sur **vscode** pour gérer les requêtes à la BDD comme "**SQLTools**" qui doit être accompagner du driver correspondant "**SQLTools MySQL/MariaDB**".

On peut aussi utiliser "**MySQL Workbench**" ou "**PHP My Admin**" pour gérer nos BDD ou générer des diagrammes de BDD.

Utilisez vos outils préférés pour réaliser les exercices du fichier :

Bieres-exercice.sql