

## 19) Union

Pour l'instant nous avons toujours vu que l'on pouvait utiliser qu'un seul SELECT pour une seule table. Sachez que nous pouvons utiliser plusieurs SELECT grâce au mot clé **UNION**.

Si par exemple j'ai deux tables d'utilisateur, une comportant toutes les femmes et une tous les hommes ou encore une comportant tous les utilisateurs dont le nom de famille va de A à M et une autre de N à Z. Si j'ai 2 sites avec la même table et que je veux récupérer les données des deux sites etc. Le Union va concaténer les 2 résultats.

On va essayer de le tester dans notre base de données. Donc nous allons d'abord créer une table identique à notre table « **users** ». Identique pour la structure de la table, mais avec différents enregistrements.

### (Exos)

- 1) Créer une nouvelle table « **users2** » dans votre base de données « **coursmysql** », en utilisant les mêmes noms de colonnes ainsi que les mêmes types que la table « **users** ». En gros une table qui a la même structure que la table « **users** ». Réfléchissez sur la manière la plus simple de le faire. (Basez-vous sur votre table **users** de base)

```
mysql> describe users2;
```

Field	Type	Null	Key	Default	Extra
id_user	int(11)	NO	PRI	NULL	auto_increment
firstname	varchar(255)	NO		NULL	
lastname	varchar(255)	NO		NULL	
sexe	enum('M','F','X','')	NO		NULL	
date_of_birth	date	NO		NULL	
city	varchar(255)	NO		NULL	
weight_kg	int(5)	NO		NULL	

7 rows in set (0.00 sec)

- 2) Une fois que votre table « **users** » et « **users2** » ont exactement la même structure, rajouter lui **ces 5** utilisateurs :

```
mysql> SELECT * FROM users2;
```

id_user	firstname	lastname	sexe	date_of_birth	city	weight_kg
1	Jessica	Biel	F	1985-11-02	Los Angeles	59
2	Mohamed	Salah	M	1989-03-10	Caire	80
3	Sadio	Mané	M	1990-05-17	Dakar	82
4	Nathan	Drake	X	2010-06-30	El Dorado	75
5	Angelina	Jolie	F	1973-09-21	Texas	55

5 rows in set (0.00 sec)

Maintenant qu'on a créé notre nouvelle table, on peut appliquer l'UNION pour afficher tous ce qu'on veut concernant ces deux tables. **Attention pour utiliser le UNION il faut exactement le même nombre de colonne demandé.**

Syntaxe :

- **SELECT \* FROM nomTable1**  
**UNION**  
**SELECT \* FROM nomTable2**

```
SELECT * FROM users UNION SELECT * FROM users2
```

☐ Tout afficher | Nombre de lignes :  Filtre les lignes:

+ Options

id_user	firstname	lastname	sexe	date_of_birth	city	weight_kg
1	James	Bond	M	2007-07-07	London	70
2	Jack	Bauer	M	2004-12-24	New-York	124
3	Lara	Croft	F	1997-08-01	Washington	63
8	Cool	Blues	X	2000-02-24	Anvers	100
9	Beyonce	Knowles	F	1980-11-20	Kinshasa	70
10	Catelyne	Jenner	X	1970-12-10	Bruxelles	86
11	Alicia	Keys	F	1987-11-18	Anvers	72
12	Hakima	Darmouch	F	1982-01-01	Bruxelles	76
13	Kevin	Debruyne	M	1990-07-10	Anvers	79
14	Dark	Vador	M	2021-11-15	Tetouane	96
16	Mario	Bros	X	1980-11-20	DreamLand	78
1	Jessica	Biel	F	1985-11-02	Los Angeles	59
2	Mohamed	Salah	M	1989-03-10	Caire	80
3	Sadio	Mané	M	1990-05-17	Dakar	82
4	Nathan	Drake	X	2010-06-30	El Dorado	75
5	Angelina	Jolie	F	1973-09-21	Texas	55

Nous voyons bien qu'il a su afficher les deux tables. Ils ont le même nombre de colonnes.  
J'insiste **il faut le même nombre de colonne.**

Il y a 3 règles :

- **Le même nombre de colonne**
- **Le même ordre pour la cohérence**
- **Le même type de données pour la cohérence**

**A noter** qu'il ne va pas récupérer de doublons d'enregistrement parfaitement identique.  
Mais si on veut récupérer tous les enregistrements même les doublons c'est avec le **UNION ALL**.

**(Exos)**

- 3) Affichez-moi tous les prénoms et noms des hommes dans les tables users et users2.
- 4) Affichez-moi tous les utilisateurs des 2 tables qui sont né avant 1990.
- 5) Affichez-moi tous les utilisateurs des 2 tables trié par leurs prénoms de manière croissante. Que remarquez-vous ?
- 6) Affichez-moi les 5 premières femmes des 2 tables trié par leurs prénoms de manière décroissante.

Vous l'aurez remarqué que tous ce qui concerne le tri, le regroupement ou encore le LIMIT, il faudra le mettre à la toute fin du deuxième select, par ce que je le redis encore tous ces éléments ne sont pas des conditions. Ce sont des choses qui attendent d'avoir le résultat du selecte plus la condition where, ensuite en fonction de ce résultat il ordonne ou il LIMIT ou il regroupe etc.

## **20) Les clés étrangères**

Nous allons terminer ce cours avec les jointures en SQL. La jointure va permettre différente relation entre les tables de vos bases de données. Par exemple une relation entre une table utilisateur et une table article, ou chaque utilisateur peut avoir créer 0 à plusieurs articles.

Mais avant de parler de la jointure, il faut tout d'abord que je vous explique le concept de **clé étrangère**. Jusqu'ici nous avons vu les clés primaires qui était principalement l'identifiant(**id**). Donc pour chaque table nous avons la clé primaire. Normalement si vous avez bien suivi le cours, dans le PDF part 1, à la toute fin du PDF (page 15), ça se terminait par un exercice ou je vous demandais de créer une table « **articles** ». Ensuite dans le PDF part 2 (page 7), il y avait un exercice qui nous demandais de modifier la table « **articles** » en lui rajoutant entre autres un identifiant et autre colonne.

### **(Exos)**

- 7) Reprenez votre table articles :
  - Ceux qui ne l'ont pas, créez la en reprenant l'exercice du PDF part 1 pages 15, ensuite reprenez les exercices du PDF part 2 pages 7 pour pouvoir la modifier. Vous pouvez le faire via l'interface graphique PhpMyAdmin si vous n'y arrivez pas pour les types des dates (timestamp, current\_timestamp ect).

```
mysql> describe articles;
```

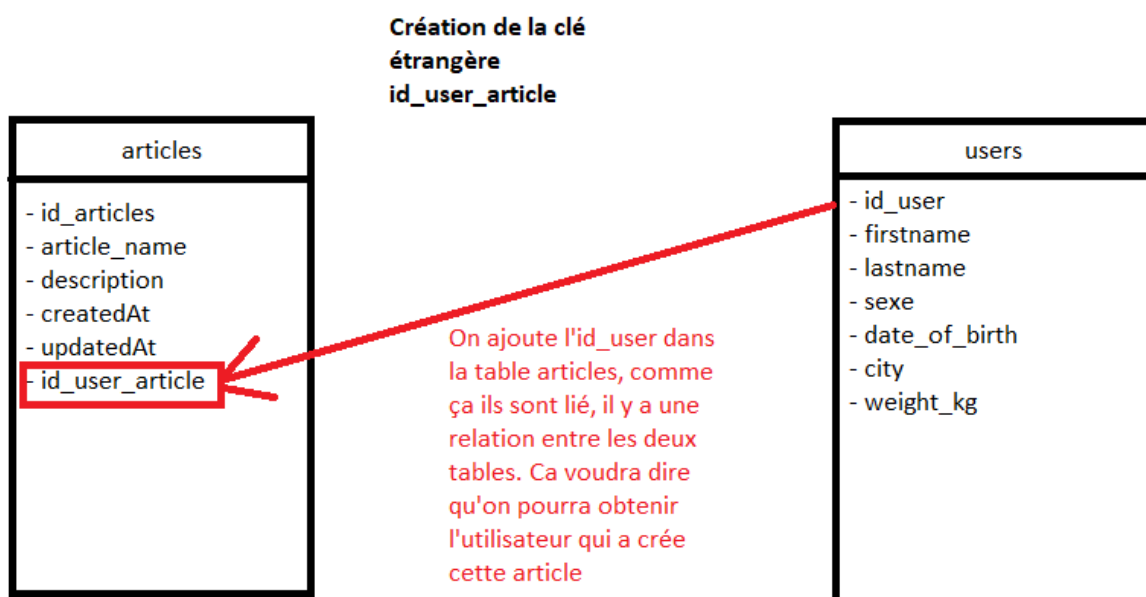
Field	Type	Null	Key	Default	Extra
id_article	int(11)	NO	PRI	NULL	auto_increment
article_name	varchar(255)	NO	UNI	NULL	
description	text	NO		NULL	
createdAt	timestamp	NO		CURRENT_TIMESTAMP	
updatedAt	timestamp	NO		CURRENT_TIMESTAMP	on update CURRENT_TIMESTAMP

5 rows in set (0.01 sec)

8) Je vais vous fournir le code pour remplir votre table articles, il ne fonctionnera que si vous avez la même structure de table de l'exercice 7.

Maintenant que tout le monde à la même table « **articles** » on va lui rajouter une clé étrangère.

La clé étrangère fonctionne comme ceci :



Donc pour faire en sorte qu'il y ait une relation entre deux tables, il faut un même élément présent dans ces 2 tables. Ici ce sera l'identifiant de l'utilisateur. Comme ça si on veut tous les articles qu'un utilisateur a publiés, il suffira de chercher dans la table articles, tous les articles qui ont l'identifiant de cet utilisateur. Et le contraire aussi est possible, si on veut savoir qui a publié un article, il suffit d'aller voir l'identifiant qui correspondra.

**Pour résumer, les clés étrangères sont des clés primaires venant d'autre table.**

On va rajouter cette fameuse clé étrangère à notre table articles. Etant donné que nos tables existent déjà et ont déjà des enregistrements ça sera un tout petit peu plus long :

- **ALTER TABLE articles ADD id\_user\_article INT NOT NULL ;**

Si vous cliquez sur votre base de données, vous pourrez voir si vos tables sont en InnoDB, c'est nécessaire pour utiliser correctement les clés étrangères. Si ce n'est pas le cas faites cette commande :

- **ALTER TABLE nomDeLaTable ENGINE=InnoDB;**

Table: coursmysql

Table	Action	Lignes	Type	Interclassement	Taille	Perte
articles	Parcourir Structure Rechercher Insérer Vider Supprimer	18	InnoDB	utf8_general_ci	80,0 kio	-
users	Parcourir Structure Rechercher Insérer Vider Supprimer	11	InnoDB	utf8_general_ci	16,0 kio	-
users2	Parcourir Structure Rechercher Insérer Vider Supprimer	5	InnoDB	utf8_general_ci	16,0 kio	-
<b>3 tables</b>	<b>Somme</b>	<b>34</b>	<b>MyISAM</b>	<b>utf8_general_ci</b>	<b>112,0 kio</b>	<b>0</b>

Ensuite il faudra faire la commande qui permet de rajouter des clés étrangères pour des tables qui ont des enregistrements (redémarrer MySQL dans CMD après avoir fait ça) :

- **SET GLOBAL FOREIGN\_KEY\_CHECKS=0;**

Une fois ces étapes passées, je vais vous montrer comment ajouter une contrainte de clé étrangère via PhpMyAdmin et son équivalent en commande SQL bien entendu.

Sur PhpMyAdmin vous pouvez cliquer sur votre table « **articles** » ensuite dans structure, puis vue relationnelle, ensuite ajouter une contrainte qu'on va appeler « **FK\_User\_Article** » pour **Foreign Key** (clé étrangère), on laisse **RESTRICT** pour le **ON DELETE** et **ON UPDATE**. Ensuite on choisit la colonne qu'on vient de créer donc « **id\_user\_article** », puis on sélectionne la base de données là où est notre table « **users** » avec sa clé primaire « **id\_user** » :

Table: articles

Contraintes de clé étrangère

Actions	Propriétés de la contrainte	Colonne	Contrainte de clé étrangère (INNODB)		
			Base de données	Table	Colonne
FK	ON DELETE: RESTRICT ON UPDATE: RESTRICT	id_user_article	coursmysql	users	id_user

+ Ajouter une contrainte

Aperçu SQL Enregistrer

Et là vous verrez que maintenant votre clé étrangère de la table « **articles** » est liée avec la clé primaire de la table « **users** ».

Tout ce qu'on vient de faire, en ligne de commande SQL il suffit de faire :

- **ALTER TABLE articles ADD CONSTRAINT FK\_user\_article FOREIGN KEY (id\_user\_article) REFERENCES users(id\_user) ON DELETE RESTRICT ON UPDATE RESTRICT;**

Affichez votre table « **articles** », vous verrez que si on clique sur le 0 d'un enregistrement de la colonne **id\_user\_article**, il vous renvoie directement vers la table « **users** ».

	id_article	article_name	description	createdAt	updatedAt	id_user_article
1	1	CAN : Mohamed Salah, le parfait trouble-fête ?	C'est tout un pays qui a retenu son souffle. Ou pl...	2022-02-03 09:23:34	2022-02-03 13:22:01	0
2	2	Cinq questions pour comprendre l'affaire Benjamin ...	Mercredi, Benjamin Mendy était convoqué pour une n...	2022-02-03 09:23:34	2022-02-03 09:24:47	0
3	3	Aubameyang au FC Barcelone ... Quand un club «dése...	Son éclatant sourire, sur des photos prises en ple...	2022-02-03 09:27:45	2022-02-03 09:27:45	0

Lien absent !

On va modifier un article via PhpMyAdmin, pour lui donner un utilisateur existant, il suffit de cliquer sur éditer :

The screenshot shows the 'Edit Record' interface in PhpMyAdmin for the 'articles' table. The form includes fields for 'id\_article', 'article\_name', 'description', 'createdAt', 'updatedAt', and 'id\_user\_article'. A dropdown menu for 'id\_user\_article' is open, displaying a list of users with their IDs. The user 'Lara' with ID 3 is highlighted. The 'Exécuter' (Execute) button is located at the bottom right of the form and is highlighted with a red box.

J'ai choisi Lara qui a comme id 3. Voici l'équivalent en ligne de commande SQL :

```
- UPDATE articles SET id_user_article = 3 WHERE articles.id_article = 1;
```

Cette fois ci on verra bien, en allant dans votre table « articles », qui est l'id\_user\_article

id_user_article	
2	3
1	1
34	10

Pareil pour James Bond qui est juste en bas avec comme id\_user\_article 1 etc. Maintenant si vous cliquez sur le numéro il vous renverra dans la table « users » en vous montrant l'utilisateur complet.

The screenshot shows the 'Table: users' view in PhpMyAdmin. It displays the SQL query used to retrieve the user: `SELECT * FROM `coursmysql`.`users` WHERE `id_user` = 3`. Below the query, the resulting record is shown in a table format.

id_user	firstname	lastname	sexe	date_of_birth	city	weight_kg
3	Lara	Croft	F	1997-08-01	Washington	63

Voilà ce fut long mais maintenant nos 2 tables sont liées. On verra par après avec Symfony qu'on n'aura pas besoin de faire tout cela.

Tout ce qu'on a fait, on sait le faire uniquement en ligne de commande.

**A noter**, si vous vous rappelez bien, dans le Contrôle à livre ouvert à la question 11 bonus, j'avais demandé de m'afficher une requête qui prenait en compte 2 tables. La question c'était :

- Afficher les noms des clients et la date de réservations, ayant réservé en 2020.

Donc allez chercher les infos dans 2 tables différentes, voici un résultat possible :

```
Select C.Nom, R.date_arr From Reservations R, Clients C Where (R.Date_arr > "2019/12/31" and R.date_arr < "2021/01/01") And R.Num_Client = C.Num_Client
```

☐ Profilage [Éditer en ligne] [ Éditer ] [ Expliquer SQL ] [ Créer le code source PHP ] [ A ]

☐ Tout afficher | Nombre de lignes : 25 ▼ Filtrer les lignes: Chercher dans cette table

+ Options

Nom	date_arr
Deschamps	2020-10-11
Cfitech	2020-12-29

Ici dans la sélection de la table (**FROM**), je mets la table **Clients** et la table **Réservations**, je leur donne ensuite un alias **C** pour **clients** et **R** pour **réservations**, comme ça je peux écrire le nom de la table en utilisant juste une lettre suivi d'un point et du nom d'un champ de cette table (exemple **C.Nom**).

La partie importante c'est **R.Num\_Client = C.Num\_Client**, ça veut dire qu'on a les mêmes numéros de client dans la table **Réservation** et dans la table **Clients**. **R.Num\_Client** est en gros un identifiant venant d'une autre table. C'est grâce à ce lien qu'on a eu la possibilité de lier ces deux tables.

### (Exos)

- 9) Vous vous souvenez que nous avons créé une table « users2 » avec les mêmes propriétés que la table « users ». Je veux que vous créiez une table « articles2 » avec la même structure que la table « articles ». Une fois fait, je veux que vous ajoutiez une clé étrangère dans la table « articles2 » qui sera relié avec la clé primaire de la table « users2 ». Reprenez les étapes une par une, **n'oubliez pas que vous pouvez vous baser sur votre table existante.**
- 10) Une fois votre table « articles2 » relié à votre table « users2 », rajoutez 5 articles et liez-les à chaque utilisateur. En gros chaque utilisateur sera relié à un article.
- 11) Vérifiez bien avec PhpMyAdmin que quand vous êtes dans votre table « articles2 » et que vous cliquez sur le numéro de id\_user\_article, il vous renvoi bien vers la table « users2 » en montrant l'utilisateur correspondant à cet id.

Tout ce qu'on vient de faire, c'est pour pouvoir utiliser les jointures.