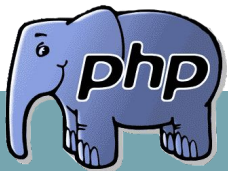


PHP



PHP

SQL, les jointures.



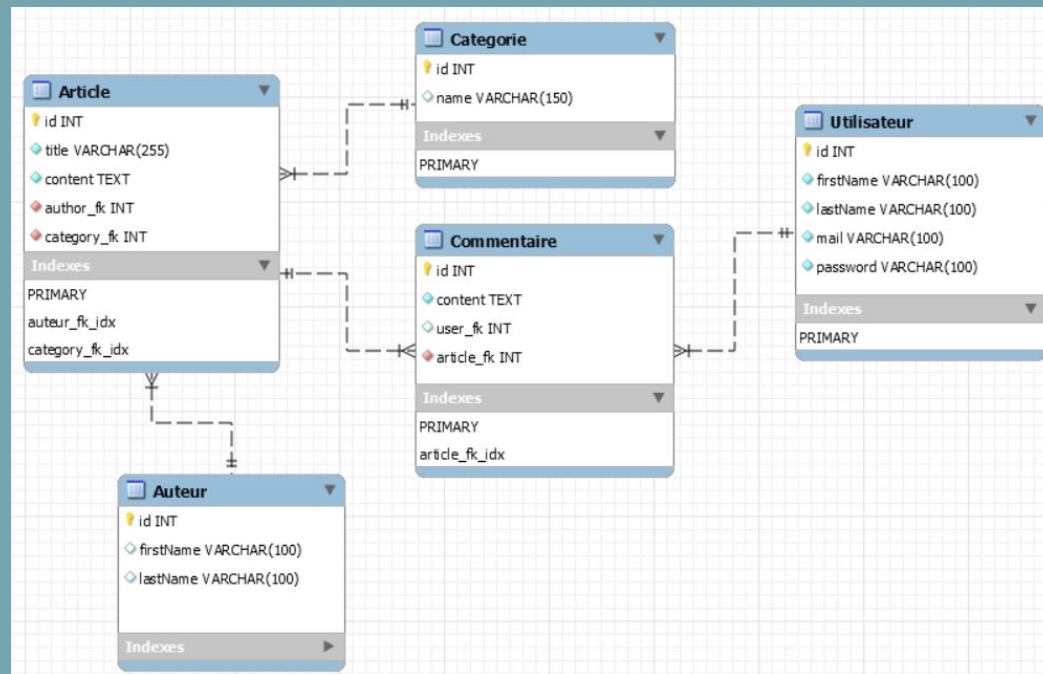
PHP - SQL, les jointures.

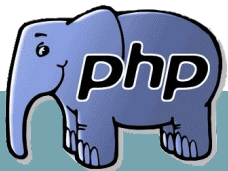
L'intérêt des bases de données est qu'on va pouvoir manipuler des données dans plusieurs tables à la fois, c'est possible grâce aux *jointures* et à la clause *JOIN*.

Si votre base de données est bien faite, vous devriez avoir de nombreuses tables qui contiennent chacune des informations liées entre elles par des relations. Pour un blog, vous devriez avoir par exemple :

- Une table article qui contient tous les articles.
- Une table catégorie qui contient les catégories de vos articles.
- Une table commentaire pour les commentaires utilisateurs.
- Une table auteur qui contient tous les auteurs des articles.

Voici ce que ça pourrait donner en ayant utilisé Mysql Workbench pour modéliser cette base de données.





PHP - SQL, les jointures.

Cette segmentation en petites tables permet de mieux vous organiser et rend l'accès à vos données plus simple et plus souple. Un autre avantage est que si un auteur écrit deux billets de blog, on ne devra pas à chaque fois stocker les informations de l'auteur dans la table article puisque nous avons déjà ces informations au niveau de la table auteur, l'id de l'auteur en provenance de la table auteur suffira.

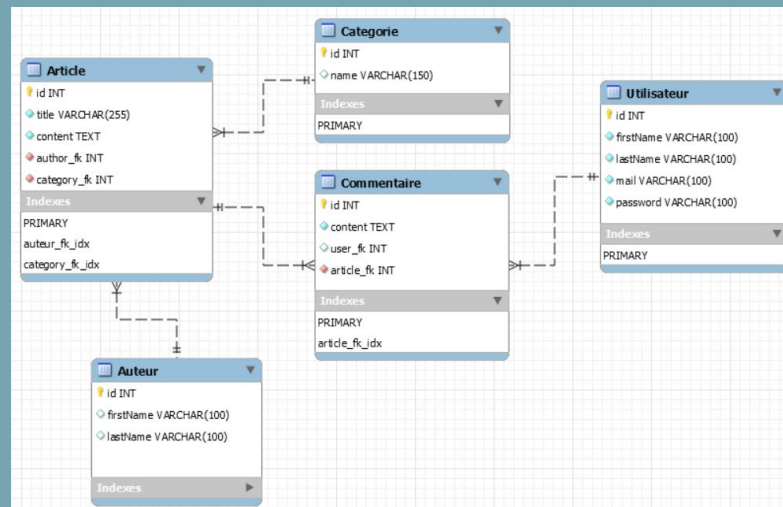
On veillera donc à être en mesure de réutiliser un maximum d'informations en séparant au mieux les tables / données, de cette manière, on pourra gérer nos données par une simple relation (relation entre entités).

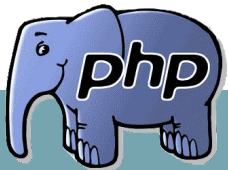
Dans ce cadre, il va donc être très intéressant de pouvoir aller rechercher de l'information dans plusieurs tables à la fois de manière à pouvoir effectuer des requêtes qui comportent des données pertinentes en retour.

Retenez donc cette manière de procéder pour structurer vos données, c'est non seulement une bonne pratique, mais ça vous aidera également à créer des applications robustes et moins gourmandes en terme de stockage de données.

Avec ce type de structure, il va être intéressant de pouvoir sélectionner des informations dans plusieurs tables en une seule fois plutôt que de créer plusieurs requêtes.

Les jointures vous permettront d'arriver à ce résultat, elles vous permettront de sélectionner des champs se trouvant dans différentes table afin de vous retourner un résultat cohérent, comme si les données provenaient de la même table.



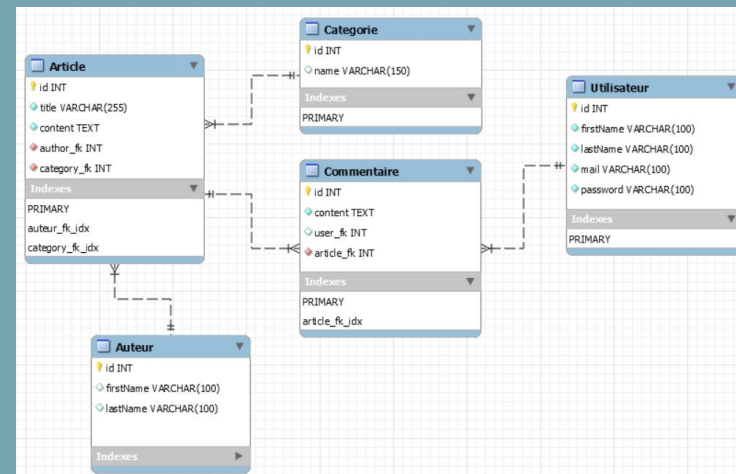


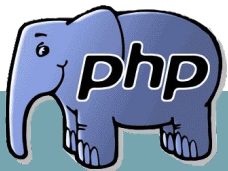
PHP - SQL, les jointures, plus en détail.

Pour pouvoir effectuer une jointure, nous allons utiliser la clause JOIN de manière à combiner des entrées entre plusieurs tables et à retourner en résultat toutes les données de toutes les colonnes sélectionnées dans toutes les tables sélectionnées.

Pour pouvoir “lier” des tables entre elles, on va devoir utiliser un liant, ou plutôt un point commun entre les tables sélectionnées pour récupérer de l'information. Ce point commun sera une colonne contenant les mêmes données d'une table à l'autre. Tiens tiens, n'utilisons pas déjà des colonnes contenant l'id d'une donnée d'une autre table, ces fameuses clés étrangères ??? Retenez qu'on utilisera toujours une colonne contenant une valeur unique pour réaliser nos jointures.

C'est bien là, et j'insiste sur ce point, que réside toute la difficulté de bien construire sa base de données : il faut bien réfléchir aux différentes situations que l'on pourrait rencontrer et aux opérations qu'on va vouloir effectuer dessus ! D'où l'intérêt d'utiliser MySql Workbench et de prendre son temps pour penser la structure de la base de données.





PHP - SQL, les jointures et alias.

Avec les jointures, nous allons travailler sur plusieurs tables et avec plus de colonnes, pour ne pas nous perdre dans toutes ces colonnes on pourra utiliser des alias de manière à mieux visualiser à quelle table appartient telle ou telle colonne.

Les alias permettent de renommer temporairement une table ou une colonne, pour en créer un, on utilise le mot clé AS suivi du nom temporaire à appliquer.

Les alias vont ainsi nous permettre de renommer temporairement des tables ou des colonnes afin de les rendre plus faciles à manipuler et de rendre nos requêtes SQL plus lisibles lorsque nous utilisons les jointures.

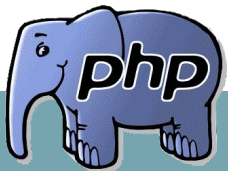
```
try{
    $conn = new PDO( dsn: "mysql:host=$server;dbname=$db;charset=utf8", $user, $password);
    $conn->setAttribute( attribute: PDO::ATTR_ERRMODE, value: PDO::ERRMODE_EXCEPTION);
    $conn->setAttribute( attribute: PDO::ATTR_DEFAULT_FETCH_MODE, value: PDO::FETCH_ASSOC);
}
catch(PDOException $e){
    // Die permet de stoper le script PHP si une erreur est survenue.
    die("Une erreur est survenue: " . $e->getMessage());
}
```

```
$request = $conn->prepare( query: "
    SELECT id AS i, title AS t, content AS c
    FROM Article AS a WHERE id <= 5
");
```

```
$request->execute();
echo "<pre>";
print_r($request->fetchAll());
echo "</pre>";
```

Les alias ne sont pas utiles dans cet exemple, mais ils le seront lorsque nous manipulons plusieurs tables.

```
Array
(
    [0] => Array
        (
            [i] => 1
            [t] => Les jointures en SQL.
            [c] => Mon super article qui parlera des jointures en SQL blablabla...
        )
)
```



PHP - SQL, les jointures et alias.

Il existe plusieurs types de jointures mais seules 3 vous seront utiles à ce stade :







- L'INNER JOIN (jointure interne)
- Le LEFT JOIN (jointure externe, à gauche)
- Le RIGHT JOIN (jointure externe à droite)

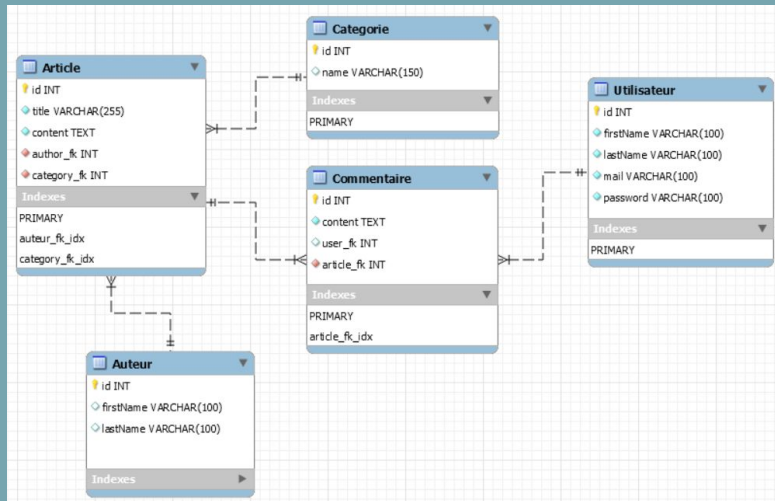
J'ai ajouté quelques données, nous allons nous concentrer sur les tables Article et Catégorie

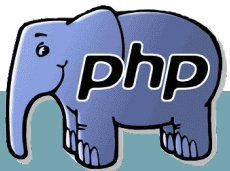
Table des Catégories :

id	name
1	SQL
2	PHP
3	JavaScript
4	HTML
5	CSS
6	jQuery

Table des Articles:

+ Options											
<div>↩️⤵️↪️</div>			id	title	content	▼	author_fk	category_fk			
<input type="checkbox"/>		Éditer		Copier		Supprimer	1	Les jointures en SQL.	Mon super article qui parlera des jointures en SQL...	2	1
<input type="checkbox"/>		Éditer		Copier		Supprimer	2	LeSQL facile !	Le SQL c'est cool et c'est facile !	1	1





PHP - SQL, INNER JOIN.

INNER JOIN est une jointure interne, elle nous permet de sélectionner UNIQUEMENT les données qui ont une valeur identique dans les deux tables. Le résultat peut être comparé à une nouvelle table temporaire contenant toutes les informations demandées.

```
$request = $conn->prepare( query: "
    SELECT Article.title, Article.content, Auteur.firstName, Auteur.lastName
    FROM Article
    INNER JOIN Auteur ON Article.author_fk = Auteur.id
");
```

```
$request->execute();
echo "<pre>";
print_r($request->fetchAll());
echo "</pre>";
```

La même chose en utilisant des alias:

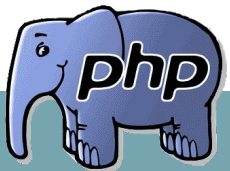
```
$request = $conn->prepare( query: "
    SELECT ar.title, ar.content, au.firstName, au.lastName
    FROM Article as ar
    INNER JOIN Auteur as au ON ar.author_fk = au.id
");
```

```
$request->execute();
echo "<pre>";
print_r($request->fetchAll());
echo "</pre>";
```

```
Array
(
    [0] => Array
        (
            [title] => Les jointures en SQL.
            [content] => Mon super article qui parlera des jointures en SQL blablabla...
            [firstName] => Jane
            [lastName] => Doe
        )

    [1] => Array
        (
            [title] => LeSQL facile !
            [content] => Le SQL c'est cool et c'est facile !
            [firstName] => John
            [lastName] => Doe
        )

)
```



PHP - SQL, INNER JOIN.

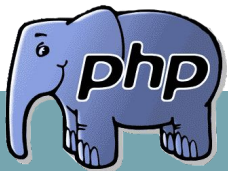
Ici, on sélectionne toutes les données des colonnes "title" et "content" de la table "Article" ET toutes les données des colonnes "firstName" et "lastName" de la table « Auteur » si la colonne "id" de la table "Article" trouve un équivalent dans la colonne "author_fk" de la table "Auteur".

```
$request = $conn->prepare( query: "  
    SELECT Article.title, Article.content, Auteur.firstName, Auteur.lastName  
    FROM Article  
    INNER JOIN Auteur ON Article.author_fk = Auteur.id  
");
```

Les Articles qui n'ont pas d'auteur ET les Auteurs qui n'ont pas écrit d'article seront ignorés. Ici, pour ne pas que nous nous trompions et pour ne pas que SQL se trompe (ambiguïté), on utilisera le nom de la table a chaque fois avant la colonne à récupérer.

On pourrait se contenter de n'effectuer qu'une jointure entre deux tables, mais sachez qu'il est tout à fait possible d'ajouter encore plus d'informations (colonnes) en ajoutant simplement un autre INNER JOIN à notre requête.

Sur la page suivante, nous allons récupérer toutes les informations concernant un Article, y compris l'auteur et la catégorie qui se trouvent dans leurs propres tables.



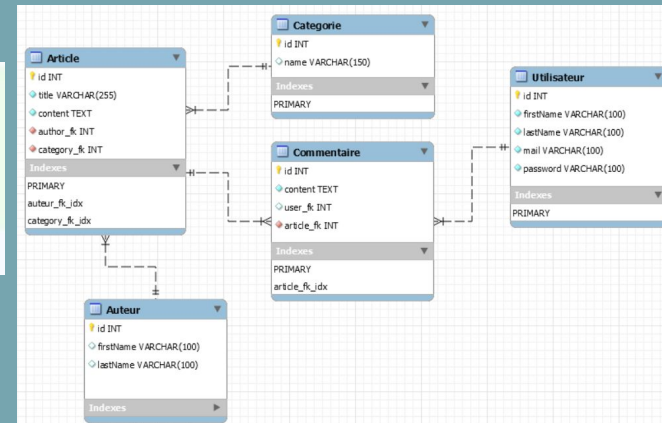
PHP - SQL, INNER JOIN.

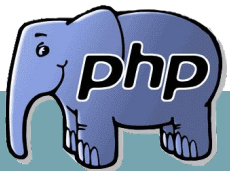
```
$request = $conn->prepare( query: "  
    SELECT Article.title, Article.content, Auteur.firstName, Auteur.lastName, Categorie.name  
    FROM Article  
    INNER JOIN Auteur ON Article.author_fk = Auteur.id  
    INNER JOIN Categorie ON Article.category_fk = Categorie.id  
");
```

Dans mon exemple, je récupère toutes les informations d'un Article, y compris la catégorie et l'auteur qui se trouvent pourtant dans une autre table, tout ça grâce aux clés étrangères et à la jointure.

Dans le code du dessus, la colonne contenant la catégorie se nomme "name", hors ici ce n'est pas pertinent, rien ne nous empêche d'utiliser les alias de manière à se retrouver avec des noms de colonnes plus pertinents :

```
$request = $conn->prepare( query: "  
    SELECT Article.title, Article.content, Auteur.firstName, Auteur.lastName, Categorie.name as categorie  
    FROM Article  
    INNER JOIN Auteur ON Article.author_fk = Auteur.id  
    INNER JOIN Categorie ON Article.category_fk = Categorie.id  
");
```



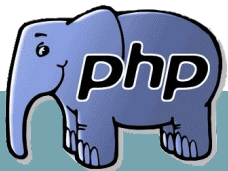


PHP - SQL, LEFT JOIN.

```
Array
(
    [0] => Array
        (
            [title] => Les jointures en SQL.
            [content] => Mon super article qui parlera des jointures en SQL blablabla...
            [firstName] => Jane
            [lastName] => Doe
            [categorie] => SQL
        )

    [1] => Array
        (
            [title] => LeSQL facile !
            [content] => Le SQL c'est cool et c'est facile !
            [firstName] => John
            [lastName] => Doe
            [categorie] => SQL
        )

)
```



PHP - SQL, LEFT JOIN.

Le LEFT JOIN est un type de jointures externes, il permet de récupérer toutes les données de la table de gauche (table de départ) ET les données de la table de droite (table sur laquelle on fait la jointure) qui ont une correspondance dans la table de gauche.

Avec ce type de requête, on va pouvoir récupérer tous les noms et prénoms des auteurs de la table "Auteur" et chaque titre d'article liés à un auteur de notre table "Article", s'il y en a un. Si l'auteur n'a pas écrit d'article, un résultat vide pour "title" est retourné.

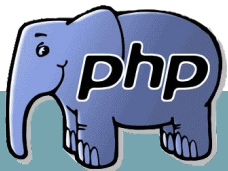
Le contenu de la table se trouvant à gauche (Auteur) sera toujours retourné :

```
$request = $conn->prepare( query: "  
    SELECT Auteur.firstName, Auteur.lastName, Article.title  
    FROM Auteur  
    LEFT JOIN Article ON Auteur.id = Article.author_fk  
");
```

Avec les alias:

```
$request = $conn->prepare( query: "  
    SELECT au.firstName, au.lastName, au.title  
    FROM Auteur AS au  
    LEFT JOIN Article AS ar ON au.id = ar.author_fk  
");
```

```
Array  
(  
    [0] => Array  
        (  
            [firstName] => Jane  
            [lastName] => Doe  
            [title] => Les jointures en SQL.  
        )  
    [1] => Array  
        (  
            [firstName] => John  
            [lastName] => Doe  
            [title] => LeSQL facile !  
        )  
    [2] => Array  
        (  
            [firstName] => Jane  
            [lastName] => Doe  
            [title] => Deuxième article de Jane  
        )  
    [3] => Array  
        (  
            [firstName] => Sarah  
            [lastName] => Conor  
            [title] =>  
        )  
    [4] => Array  
        (  
            [firstName] => Luc  
            [lastName] => Skywalker  
            [title] =>  
        )  
)
```



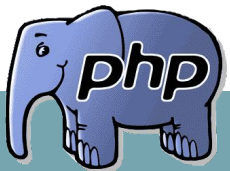
PHP - SQL, LEFT JOIN.

Tout comme pour inner join, il vous est possible d'enchaîner les LEFT JOIN de manière à récupérer plus d'information (sur 3 tables par exemple) :

```
$request = $conn->prepare( query: "  
    SELECT Article.title, Auteur.firstName, Auteur.lastName, Categorie.name  
    FROM Article  
    LEFT JOIN Auteur ON Auteur.id = Article.author_fk  
    LEFT JOIN Categorie ON Categorie.id = Article.category_fk  
");
```

On pourrait croire que ce résultat est identique à INNER JOIN, mais c'est uniquement parce que je ne dispose d'aucun article qui n'ai pas de catégorie ou qui n'ait pas d'auteur. Si ça avait été le cas (pas de catégorie et / ou pas d'auteur, l'article aurait quand même été retourné, contrairement à INNER JOIN

```
Array  
(  
    [0] => Array  
        (  
            [title] => Les jointures en SQL.  
            [firstName] => Jane  
            [lastName] => Doe  
            [name] => SQL  
        )  
    [1] => Array  
        (  
            [title] => LeSQL facile !  
            [firstName] => John  
            [lastName] => Doe  
            [name] => SQL  
        )  
    [2] => Array  
        (  
            [title] => Deuxième article de Jane  
            [firstName] => Jane  
            [lastName] => Doe  
            [name] => CSS  
        )  
)
```



PHP - SQL, RIGHT JOIN.

Le RIGHT JOIN est un autre type de jointures externes, il fonctionne comme LEFT JOIN sauf que la table de référence (table sur laquelle on effectue la jointure) est la table de droite dans la requête SQL. Toutes les données de la table de droite seront récupérées tandis que seules les entrées de la table de gauche (table de départ) qui vont satisfaire à la condition de jointure seront sélectionnées (JOIN ... ON ... = condition de jointure).

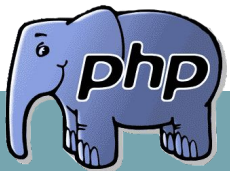
```
$request = $conn->prepare( query: "  
    SELECT Article.title, Auteur.firstName, Auteur.lastName  
    FROM Article  
    RIGHT JOIN Auteur ON Auteur.id = Article.author_fk  
");
```

On fait la même chose que pour le LEFT JOIN, sauf qu'ici avec RIGHT JOIN, on demande à ce que tous les résultats de la table "Auteur" soient retournés, même si l'auteur ne figure pas dans la table "Article"

Dans l'exemple, Jane dispose de deux articles, hors il serait plus propre de réunir les articles de Jane de manière à pouvoir les retrouver les uns en dessous des autres pour par exemple effectuer un traitement ou tout simplement les regrouper pour affichage.

C'est ce que nous verrons en page suivante.

```
Array  
(  
    [0] => Array  
        (  
            [title] => Les jointures en SQL.  
            [firstName] => Jane  
            [lastName] => Doe  
        )  
    [1] => Array  
        (  
            [title] => LeSQL facile !  
            [firstName] => John  
            [lastName] => Doe  
        )  
    [2] => Array  
        (  
            [title] => Deuxième article de Jane  
            [firstName] => Jane  
            [lastName] => Doe  
        )  
    [3] => Array  
        (  
            [title] =>  
            [firstName] => Sarah  
            [lastName] => Conor  
        )  
    [4] => Array  
        (  
            [title] =>  
            [firstName] => Luc  
            [lastName] => Skywalker  
        )  
)
```



PHP - SQL, JOIN et ORDER BY.

ORDER BY vous permettra de trier les résultats de façon naturelle, de cette manière, tous les articles de John seront regroupés entre eux dans le tableau de résultat, et les articles écrits par Jane seront également regroupés entre eux :

```
$request = $conn->prepare( query: "
    SELECT Article.title, Auteur.firstName, Auteur.lastName
    FROM Article
    RIGHT JOIN Auteur ON Auteur.id = Article.author_fk
    ORDER BY Auteur.firstName
");
```

On pourrait aussi demander un tri en premier lieu sur le prénom, et en deuxième lieu sur le nom :

```
$request = $conn->prepare( query: "
    SELECT Article.title, Auteur.firstName, Auteur.lastName
    FROM Article
    RIGHT JOIN Auteur ON Auteur.id = Article.author_fk
    ORDER BY Auteur.firstName, Auteur.lastName
");
```

```
Array
(
    [0] => Array
        (
            [title] => Deuxième article de Jane
            [firstName] => Jane
            [lastName] => Doe
        )

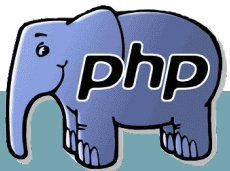
    [1] => Array
        (
            [title] => Les jointures en SQL.
            [firstName] => Jane
            [lastName] => Doe
        )

    [2] => Array
        (
            [title] => LeSQL facile !
            [firstName] => John
            [lastName] => Doe
        )

    [3] => Array
        (
            [title] =>
            [firstName] => Luc
            [lastName] => Skywalker
        )

    [4] => Array
        (
            [title] =>
            [firstName] => Sarah
            [lastName] => Conor
        )

)
```



PHP - SQL, JOIN et la clause WHERE.

Pour l'instant, nous nous sommes contentés de récupérer toutes les informations d'une table à gauche ou à droite, sachez que vous pouvez utiliser la clause WHERE dans votre SELECT comme vous en avez l'habitude !

```
$request = $conn->prepare( query: "
    SELECT Article.title, Auteur.firstName, Auteur.lastName
    FROM Article
    RIGHT JOIN Auteur ON Auteur.id = Article.author_fk
    WHERE Auteur.lastName = :name
    ORDER BY Auteur.firstName
");
$request->bindValue( param: ':name', value: 'Doe');
$request->execute();
```

```
Array
(
    [0] => Array
        (
            [title] => Deuxième article de Jane
            [firstName] => Jane
            [lastName] => Doe
        )
    [1] => Array
        (
            [title] => Les jointures en SQL.
            [firstName] => Jane
            [lastName] => Doe
        )
    [2] => Array
        (
            [title] => LeSQL facile !
            [firstName] => John
            [lastName] => Doe
        )
)
```