

Les jointures SQL

Mise en place

Créez une table `article` :

```
CREATE TABLE `test`.`article` (  
  `idarticle` INT NOT NULL AUTO_INCREMENT,  
  `title` VARCHAR(255) NULL,  
  `content` LONGTEXT NULL,  
  `author` INT NULL,  
  PRIMARY KEY (`idarticle`));
```

Insérez un article :

```
INSERT INTO `test`.`article`  
(`idarticle`,  
 `title`,  
 `content`,  
 `author`)  
VALUES  
(1,  
 'je suis un titre',  
 'je suis un contenu',  
 1);
```

Création d'une clé étrangère

Créez la clé étrangère dans la table `article`.

Cliquez du droit sur la table et faites `Alter table`.

Allez dans l'onglet `Foreign Keys` :

SCHEMAS

Filter objects

sys

test

Tables

article

Columns

Indexes

Foreign Keys

Triggers

user

Views

Stored Procedures

Functions

Object Info

Session

schema: test

Name: article

Schema: test

Foreign Key Name	Referenced Table	Foreign Key Columns
		ColumnReferenced Column
fk_article_user	`test`.`user`	<div><div><input type="checkbox"/></div>idarticle</div> <div><div><input type="checkbox"/></div>title</div> <div><div><input type="checkbox"/></div>content</div> <div><div><input checked="" type="checkbox"/></div>author<div><input type="checkbox"/></div>id_user</div>

Columns

Indexes

Foreign Keys

Triggers

Partitioning

Options

Ce qui se traduit par :

```
ALTER TABLE `test`.`article`  
ADD INDEX `fk_article_user_idx` (`author` ASC) VISIBLE;  
;  
ALTER TABLE `test`.`article`  
ADD CONSTRAINT `fk_article_user`  
  FOREIGN KEY (`author`)  
  REFERENCES `test`.`user` (`id_user`)
```

```
ON DELETE NO ACTION;  
ON UPDATE NO ACTION;
```

La relation entre un utilisateur et un article est de type **One-to-many**. Un auteur peut écrire beaucoup d'articles, mais un article ne peut avoir qu'un seul auteur.

Nous relierons ici la clé primaire de l'auteur (**id_user**) de la table **user**, c'est-à-dire son identifiant unique, à la colonne **author** de la table **article**.

Dans cette relation, la table **user** est appelée la table parent et la table **article** la table enfant.

Nous verrons en détail les clés étrangères dans la formation **SQL** dédiée. Retenez ici simplement l'utilisation avec **Workbench**.

Effectuer une jointure

Les jointures **SQL** permettent d'associer plusieurs tables dans une même requête.

Elles permettent de récupérer des données relationnelles situées dans plusieurs tables en une seule requête.

Nous verrons tous les types de jointure **SQL** dans la formation dédiée.

Nous allons pour le moment voir uniquement l'**INNER JOIN** qui permet de joindre deux tables en utilisant une condition appelée **prédicat de jointure**.

Cette jointure compare chaque rangée de la première table avec chaque rangée de la seconde table.

Si les valeurs des deux rangées respectent la condition spécifiée, la jointure crée une nouvelle rangée qui contient toutes les colonnes des deux tables pour la retourner.

Autrement dit, toutes les données de toutes les colonnes des deux tables sont fusionnées dans une nouvelle rangée à chaque fois que deux rangées des deux tables valident la condition.

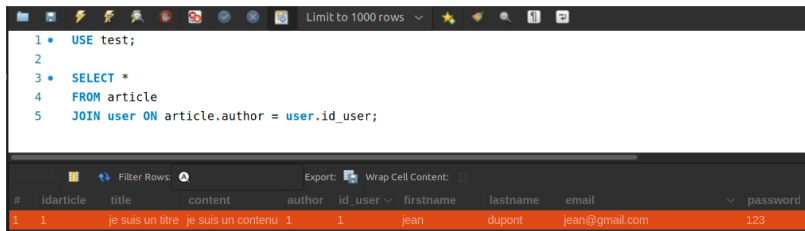
La syntaxe est :

```
SELECT colonnes  
FROM table_1  
INNER JOIN table_2 ON condition;
```

Dans notre exemple :

```
USE test;  
SELECT *  
FROM article  
JOIN user ON article.author = user.id_user;
```

Ce qui donne par exemple :



The screenshot shows a SQL IDE interface. At the top, there's a toolbar with various icons and a 'Limit to 1000 rows' dropdown. Below the toolbar, the SQL query is displayed in a text area:

```
1 • USE test;  
2  
3 • SELECT *  
4 FROM article  
5 JOIN user ON article.author = user.id_user;
```

Below the query, there's a table of results. The table has 10 columns: #, idarticle, title, content, author, id_user, firstname, lastname, email, and password. The first row of data is highlighted in orange:

#	idarticle	title	content	author	id_user	firstname	lastname	email	password
1	1	je suis un titre	je suis un contenu	1	1	jean	dupont	jean@gmail.com	123