Concevez votre site web avec PHP et MySQL

70 heures Moyenne

Mis à jour le 30/09/2019

Les jointures entre tables

MySQL permet de travailler avec plusieurs tables à la fois. Un des principaux intérêts d'une base de données est de pouvoir créer des relations entre les tables, de pouvoir les lier entre elles.

Pour le moment, nous n'avons travaillé que sur une seule table à la fois. Dans la pratique, vous aurez certainement plusieurs tables dans votre base, dont la plupart seront interconnectées. Cela vous permettra de mieux découper vos informations, d'éviter des répétitions et de rendre ainsi vos données plus faciles à gérer.

Tenez, par exemple, dans notre table `jeux_video`, on répète à chaque fois le nom du possesseur du jeu. Le mot « Patrick » est écrit de nombreuses fois dans la table. Imaginez que l'on souhaite stocker aussi son nom de famille, son adresse, son numéro de téléphone... On ne va quand même pas recopier ces informations pour chaque jeu qu'il possède ! Il est temps de créer une autre table et de la lier.

Modélisation d'une relation

Si je voulais stocker les nom, prénom et numéro de téléphone de chaque propriétaire de jeux vidéo dans notre table `jeux_video`, il n'y aurait pas d'autre solution que de dupliquer ces informations sur chaque entrée... Cependant ce serait bien trop répétitif ; regardez ce que ça donnerait sur le tableau suivant.

ID	nom	prenom	nom_possesseur	tel	console	prix	nbre_joueurs_max	con

ID	nom	prenom	nom_possesseur	tel	console	prix	nbre_joueurs_max	com
1	Super Mario Bros	Florent	Dugommier	01 44 77 21 33	NES	4	1	Un j d'ar
2	Sonic	Patrick	Lejeune	03 22 17 41 22	Megadrive	2	1	Pou mei mor
3	Zelda : ocarina of time	Florent	Dugommier	01 44 77 21 33	Nintendo 64	15	1	Un j bea com com voit de r
4	Mario Kart 64	Florent	Dugommier	01 44 77 21 33	Nintendo 64	25	4	Un jeu
5	Super Smash Bros Melee	Michel	Doussand	04 11 78 02 00	GameCube	55	4	Un j bast !

Comme vous le voyez, le nom, le prénom et le numéro de téléphone de Florent apparaissent autant de fois qu'il possède de jeux vidéo, et il en irait de même pour Patrick et Michel. Il faut à tout prix éviter ces répétitions.

Ce que je vous propose, c'est de créer une autre table, que l'on nommera par exemple `proprietaires`, qui centralisera les informations des propriétaires des jeux (tableau suivant).

ID	prenom	nom	tel
1	Florent	Dugommier	01 44 77 21 33

ID	prenom	nom	tel
2	Patrick	Lejeune	03 22 17 41 22
3	Michel	Doussand	04 11 78 02 00

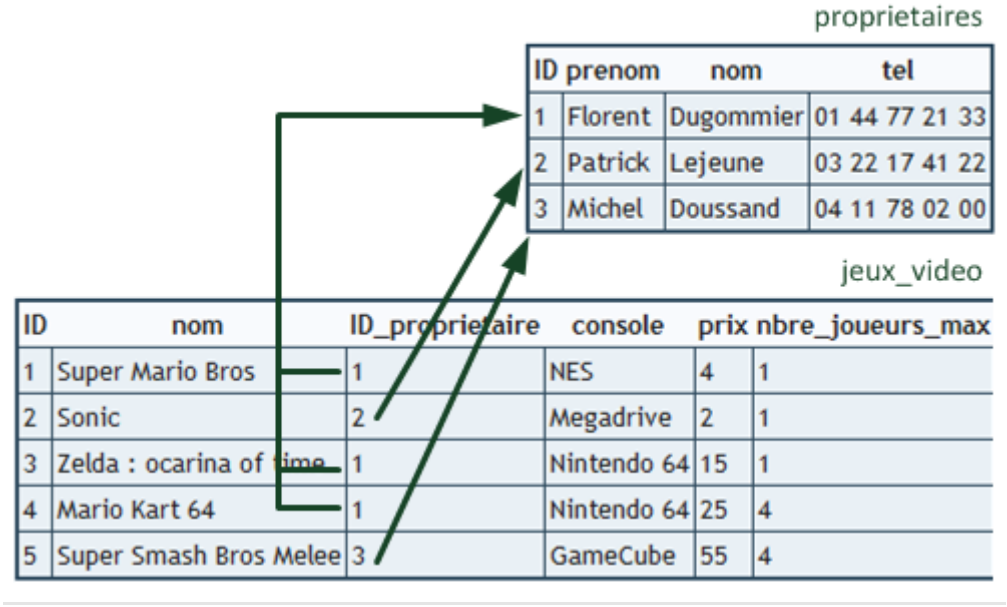
Cette table liste tous les propriétaires de jeux connus et attribue à chacun un ID. Les propriétaires n'apparaissant qu'une seule fois, il n'y a pas de doublon.

Maintenant, il faut modifier la structure de la table `jeux_video` pour faire référence aux propriétaires. Pour cela, le mieux est de créer un champ `ID_proprietaire` qui indique le numéro du propriétaire dans l'autre table (tableau suivant).

ID	nom	ID_proprietaire	console	prix	nbre_joueurs_max	commentaires
1	Super Mario Bros	1	NES	4	1	Un jeu d'anthologie !
2	Sonic	2	Megadrive	2	1	Pour moi, le meilleur jeu au monde !
3	Zelda : ocarina of time	1	Nintendo 64	15	1	Un jeu grand, beau et complet comme on en voit rarement de nos jours
4	Mario Kart 64	1	Nintendo 64	25	4	Un excellent jeu de kart !
5	Super Smash Bros Melee	3	GameCube	55	4	Un jeu de baston délirant !

Le nouveau champ `ID_proprietaire` est de type `INT`. Il permet de faire référence à une entrée précise de la table `proprietaires`.

On peut maintenant considérer que les tables sont reliées à travers ces ID de propriétaires, comme le suggère la figure suivante.



Relation entre deux tables

MySQL sait donc que l'`ID_proprietaire` no 1 dans la table `jeux_video` correspond à Florent ?

Non, il ne le sait pas. Il ne voit que des nombres et il ne fait pas la relation entre les deux tables. Il va falloir lui expliquer cette relation dans une requête SQL : on va faire ce qu'on appelle une **jointure** entre les deux tables.

Qu'est-ce qu'une jointure ?

Nous avons donc maintenant deux tables :

- `jeux_video` ;
- `proprietaires`.

Les informations sont séparées dans des tables différentes et c'est bien. Cela évite de dupliquer des informations sur le disque.

Cependant, lorsqu'on récupère la liste des jeux, si on souhaite obtenir le nom du propriétaire, il va falloir adapter la requête pour récupérer aussi les informations issues de la table `proprietaires`. Pour cela, on doit faire ce qu'on appelle une **jointure**.

Il existe plusieurs types de jointures, qui nous permettent de choisir exactement les données que l'on veut récupérer. Je vous propose d'en découvrir deux, les plus importantes :

- **les jointures internes** : elles ne sélectionnent que les données qui ont une correspondance entre les deux tables ;
- **les jointures externes** : elles sélectionnent toutes les données, même si certaines n'ont pas de correspondance dans l'autre table.

Il est important de bien comprendre la différence entre une jointure interne et une jointure externe.

Pour cela, imaginons que nous ayons une 4e personne dans la table des propriétaires, un certain Romain Vipelli, qui ne possède aucun jeu (tableau suivant).

ID	prenom	nom	tel
1	Florent	Dugommier	01 44 77 21 33
2	Patrick	Lejeune	03 22 17 41 22
3	Michel	Doussand	04 11 78 02 00
4	Romain	Vipelli	01 21 98 51 01

Romain Vipelli est référencé dans la table `proprietaires` mais il n'apparaît nulle part dans la table `jeux_video` car il ne possède aucun jeu.

Si vous récupérez les données des deux tables à l'aide :

- d'une **jointure interne** : Romain Vipelli n'apparaîtra pas dans les résultats de la requête. La jointure interne force les données d'une table à avoir une correspondance dans l'autre ;
- d'une **jointure externe** : vous aurez toutes les données de la table des propriétaires, même s'il n'y a pas de correspondance dans l'autre table des jeux vidéo ; donc Romain Vipelli, qui pourtant ne possède aucun jeu vidéo, apparaîtra.

La jointure externe est donc plus complète car elle est capable de récupérer plus d'informations, tandis que la jointure interne est plus stricte car elle ne récupère que les données qui ont une équivalence dans l'autre table.

Voici par exemple les données que l'on récupèrerait avec une jointure interne (tableau suivant) :

nom_jeu	prenom
Super Mario Bros	Florent
Sonic	Patrick
...	...

On obtient les jeux et leurs propriétaires, mais Romain qui ne possède pas de jeu n'apparaît pas du tout. En revanche, avec une jointure externe (tableau suivant) :

nom_jeu	prenom

nom_jeu	prenom
Super Mario Bros	Florent
Sonic	Patrick
...	...
`NULL`	Romain

Romain apparaît maintenant. Comme il ne possède pas de jeu, il n'y a aucun nom de jeu indiqué (`NULL`).

Nous allons maintenant voir comment réaliser ces deux types de jointures en pratique.

Les jointures internes

Une jointure interne peut être effectuée de deux façons différentes :

- à l'aide du mot-clé `WHERE` : c'est l'ancienne syntaxe, toujours utilisée aujourd'hui, qu'il faut donc connaître mais que vous devriez éviter d'utiliser si vous avez le choix ;
- à l'aide du mot-clé `JOIN` : c'est la nouvelle syntaxe qu'il est recommandé d'utiliser. Elle est plus efficace et plus lisible.

Ces deux techniques produisent exactement le même résultat, mais il faut les connaître toutes les deux. ;-)

Jointure interne avec `WHERE` (ancienne syntaxe)

Construction d'une jointure interne pas à pas

Pour réaliser ce type de jointure, on va sélectionner des champs des deux tables et indiquer le nom de ces deux tables dans la clause `FROM` :

php
```
1 SELECT nom, prenom FROM proprietaires, jeux_video
```

Cependant ça ne fonctionnera pas car ce n'est pas suffisant. En effet, le champ `nom` apparaît dans les deux tables : une fois pour le nom du propriétaire, et une autre fois pour le nom du jeu vidéo. C'est ce qu'on appelle une **colonne ambiguë** car MySQL ne sait pas s'il doit récupérer un nom de personne (comme Dugommier) ou un nom de jeu (comme Super Mario Bros). Bref, il est un peu perdu.

L'astuce consiste à marquer le nom de la table devant le nom du champ, comme ceci :

php
```
1 SELECT jeux_video.nom, proprietaires.prenom FROM proprietaires, jeux_video
```

Ainsi, on demande clairement de récupérer le nom du jeu et le prénom du propriétaire avec cette requête.

> Le champ `prenom` n'est pas ambigu, car il n'apparaît que dans la table `proprietaires`. On pourrait donc se passer d'écrire le préfixe `proprietaires` devant, mais ça ne coûte rien de le faire et c'est plus clair : on voit immédiatement en lisant la requête de quelle table est issu ce champ.

Il reste encore à lier les deux tables entre elles. En effet, les jeux et leurs propriétaires ont une correspondance via le champ `ID_proprietaire` (de la table `jeux_video`) et le champ `ID` (de la table `proprietaires`). On va indiquer cette liaison dans un `WHERE`, comme ceci :

```php
SELECT jeux_video.nom, proprietaires.prenom
FROM proprietaires, jeux_video
WHERE jeux_video.ID_proprietaire = proprietaires.ID
```

> Comme la requête devient longue, je me permets de l'écrire sur plusieurs lignes. Cette écriture est tout à fait autorisée et a l'avantage d'être plus lisible.

On indique bien que le champ `ID_proprietaire` de la table `jeux_video` correspond au champ `ID` de la table `proprietaires`. Cela établit la correspondance entre les deux tables telle qu'on l'avait définie dans le schéma suivant au début du chapitre.

Notre requête est enfin complète, vous pouvez l'essayer.

Vous devriez récupérer les données suivantes :

nom	prenom
Super Mario Bros	Florent
Sonic	Patrick
...	...

Utilisez les alias !

Nous avons appris à utiliser les alias lorsque nous avons découvert les fonctions SQL. Cela nous permettait de créer ce que j'appelais des « champs virtuels » pour représenter le résultat des fonctions.

Il est fortement conseillé d'utiliser des alias lorsqu'on fait des jointures. On peut utiliser des alias sur les noms de champs (comme on l'avait fait) :

```php
```

```
SELECT jeux_video.nom AS nom_jeu, proprietaires.prenom AS prenom_proprietaire
FROM proprietaires, jeux_video
WHERE jeux_video.ID_proprietaire = proprietaires.ID
```

On récupèrera donc deux champs : `nom_jeu` et `prenom_proprietaire`. Ces alias permettent de donner un nom plus clair aux champs que l'on récupère.

nom_jeu	prenom_proprietaire
Super Mario Bros	Florent
Sonic	Patrick
...	...

Il est également possible de donner un alias aux noms des tables, ce qui est fortement recommandé pour leur donner un nom plus court et plus facile à écrire. En général, on crée des alias de tables d'une lettre ou deux correspondant à leurs initiales, comme ceci :

php

```
SELECT j.nom AS nom_jeu, p.prenom AS prenom_proprietaire
FROM proprietaires AS p, jeux_video AS j
WHERE j.ID_proprietaire = p.ID
```

Comme vous le voyez, la table `jeux_video` a pour alias la lettre `j` et `proprietaires` la lettre `p`. On réutilise ces alias dans toute la requête, ce qui la rend plus courte à écrire (et plus lisible aussi au final).

Notez que le mot-clé `AS` est en fait facultatif, les développeurs ont tendance à l'omettre. Vous pouvez donc tout simplement le retirer de la requête :

php

```
SELECT j.nom nom_jeu, p.prenom prenom_proprietaire
FROM proprietaires p, jeux_video j
WHERE j.ID_proprietaire = p.ID
```

Jointure interne avec `JOIN` (nouvelle syntaxe)

Bien qu'il soit possible de faire une jointure interne avec un `WHERE` comme on vient de le voir, c'est une ancienne syntaxe et aujourd'hui on recommande plutôt d'utiliser `JOIN`. Il faut dire que nous étions habitués à utiliser le `WHERE` pour filtrer les données, alors que nous l'utilisons ici pour associer des tables et récupérer plus de données.

Pour éviter de confondre le `WHERE` « traditionnel » qui filtre les données et le `WHERE` de jointure que l'on vient de découvrir, on va utiliser la syntaxe `JOIN`.

Pour rappel, voici la requête qu'on utilisait avec un `WHERE` :

php

```
SELECT j.nom nom_jeu, p.prenom prenom_proprietaire
FROM proprietaires p, jeux_video j
WHERE j.ID_proprietaire = p.ID
```

Avec un `JOIN`, on écrirait cette même requête de la façon suivante :

php

```
SELECT j.nom nom_jeu, p.prenom prenom_proprietaire
FROM proprietaires p
INNER JOIN jeux_video j
ON j.ID_proprietaire = p.ID
```

Cette fois, on récupère les données depuis une table principale (ici, `proprietaires`) et on fait une jointure interne (`INNER JOIN`) avec une autre table (`jeux_video`). La liaison entre les champs est faite dans la clause `ON` un peu plus loin.

Le fonctionnement reste le même : on récupère les mêmes données que tout à l'heure avec la syntaxe `WHERE`.

Si vous voulez filtrer (`WHERE`), ordonner (`ORDER BY`) ou limiter les résultats (`LIMIT`), vous devez le faire à la fin de la requête, après le « `ON j.ID_proprietaire = p.ID` ».

Par exemple :

php

```
SELECT j.nom nom_jeu, p.prenom prenom_proprietaire
FROM proprietaires p
INNER JOIN jeux_video j
ON j.ID_proprietaire = p.ID
WHERE j.console = 'PC'
ORDER BY prix DESC
LIMIT 0, 10
```

Traduction (inspirez un grand coup avant de lire) : « Récupère le nom du jeu et le prénom du propriétaire dans les tables `proprietaires` et `jeux_video`, la liaison entre les tables se fait entre les champs `ID_proprietaire` et `ID`, prends uniquement les jeux qui tournent sur PC, trie-les par prix décroissants et ne prends que les 10 premiers. »

Il faut s'accrocher avec des requêtes de cette taille-là ! ;-)

Les jointures externes

Les jointures externes permettent de récupérer toutes les données, même celles qui n'ont pas de correspondance. On pourra ainsi obtenir Romain Vipelli dans la liste même s'il ne possède pas de jeu vidéo.

Cette fois, la seule syntaxe disponible est à base de `JOIN`. Il y a deux écritures à connaître : `LEFT JOIN` et `RIGHT JOIN`. Cela revient pratiquement au même, avec une subtile différence que nous allons voir.

`LEFT JOIN` : récupérer toute la table de gauche

Reprenons la jointure à base de `INNER JOIN` et remplaçons tout simplement `INNER` par `LEFT` :

```php
SELECT j.nom nom_jeu, p.prenom prenom_proprietaire
FROM proprietaires p
LEFT JOIN jeux_video j
ON j.ID_proprietaire = p.ID
```

`proprietaires` est appelée la « table de gauche » et `jeux_video` la « table de droite ». Le `LEFT JOIN` demande à récupérer tout le contenu de la table de gauche, donc tous les propriétaires, même si ces derniers n'ont pas d'équivalence dans la table `jeux_video` .

nom_jeu	prenom_proprietaire
Super Mario Bros	Florent
Sonic	Patrick
...	...
`NULL`	Romain

Romain apparaît désormais dans les résultats de la requête grâce à la jointure externe. Comme il ne possède aucun jeu, la colonne du nom du jeu est vide.

`RIGHT JOIN` : récupérer toute la table de droite

Le `RIGHT JOIN` demande à récupérer toutes les données de la table dite « de droite », même si celle-ci n'a pas d'équivalent dans l'autre table. Prenons la requête suivante :

```php
SELECT j.nom nom_jeu, p.prenom prenom_proprietaire
FROM proprietaires p
RIGHT JOIN jeux_video j
ON j.ID_proprietaire = p.ID
```

La table de droite est « jeux_video ». On récupèrerait donc tous les jeux, même ceux qui n'ont pas de propriétaire associé.

> Comment est-ce possible qu'un jeu n'ait pas de propriétaire associé ?

Il y a deux cas possibles :

- soit le champ `ID_proprietaire` contient une valeur qui n'a pas d'équivalent dans la table des propriétaires, par exemple « 56 » ;

- soit le champ `ID_proprietaire` vaut `NULL`, c'est-à-dire que personne ne possède ce jeu. C'est le cas notamment du jeu Bomberman dans la table que vous avez téléchargée (voir tableau suivant).

ID	nom	ID_proprietaire	console	prix	nbre_joueurs_max	commentaires
1	Super Mario Bros	1	NES	4	1	Un jeu d'anthologie !
2	Sonic	2	Megadrive	2	1	Pour moi, le meilleur jeu au monde !
3	Zelda : ocarina of time	1	Nintendo 64	15	1	Un jeu grand, beau et complet comme on en voit rarement de nos jours
4	Mario Kart 64	1	Nintendo 64	25	4	Un excellent jeu de kart !
5	Super Smash Bros Melee	3	GameCube	55	4	Un jeu de baston délirant !
...	...	...	...	...	...	...
51	Bomberman	NULL	NES	5	4	Un jeu simple et toujours aussi passionnant !

Dans ce cas, Bomberman n'appartient à personne. Avec la requête `RIGHT JOIN` que l'on vient de voir, on obtiendra toutes les lignes de la table de droite (`jeux_video`) même si elles n'ont aucun lien avec la table `proprietaires`, comme c'est le cas ici pour Bomberman.

On obtiendra donc les données exposées dans le tableau suivant.

nom_jeu	prenom_proprietaire
Super Mario Bros	Florent

nom_jeu	prenom_proprietaire
Sonic	Patrick
...	...
Bomberman	`NULL`

En résumé

- Les bases de données permettent d'associer plusieurs tables entre elles.
- Une table peut contenir les id d'une autre table ce qui permet de faire la liaison entre les deux. Par exemple, la table des jeux vidéo contient pour chaque jeu l'id de son propriétaire. Le nom et les coordonnées du propriétaire sont alors stockés dans une table à part.
- Pour rassembler les informations au moment de la requête, on effectue des **jointures**.
- On peut faire des jointures avec le mot-clé `WHERE`, mais il est recommandé d'utiliser `JOIN` qui offre plus de possibilités et qui est plus adapté.
- On distingue les jointures internes, qui retournent des données uniquement s'il y a une correspondance entre les deux tables, et les jointures externes qui retournent toutes les données même s'il n'y a pas de correspondance.

Si vous souhaitez en savoir plus sur les bases de données MySQL, je vous invite à lire le **tutoriel de Taguan entièrement dédié à MySQL**.

Que pensez-vous de ce cours ?

TP : UN BLOG AVEC DES COMMENTAIRES

QUIZ : STOCKEZ DES INFORMATIONS DANS UNE BASE DE DONNÉES

Le professeur

Mathieu Nebra

Entrepreneur à plein temps, auteur à plein temps et co-fondateur d'OpenClassrooms :o)

Découvrez aussi ce cours en...


Livre


PDF

OpenClassrooms

L'entreprise

Alternance

Forum

Blog

Nous rejoindre

Entreprises

Business

En plus

Devenez mentor

Aide et FAQ

Conditions Générales d'Utilisation

Politique de Protection des Données Personnelles

Nous contacter

Français

Télécharger dans l'App Store