Formation SQL, les jointures

m2iformation.fr

Formation SQL:

Annexe: Les jointures

Introduction

Ce support est centré sur les **jointures** qui sont une fonctionnalité avancée du **langage DQL**.

Avant d'aborder le concept de jointure, il est important d'être à l'aise avec le langage DQL, et de savoir créer BDD et tables.

Définition

En DQL, nous avons appris à récupérer les données d'**une** table. Mais il est en fait beaucoup plus fréquent que nous ayons à **récupérer les données de plusieurs tables** en même temps au sein d'une seule requête.

Les jointures nous permettent cela, on les trouve principalement sous quatres formes: **INNER JOIN, RIGHT JOIN, LEFT JOIN et FULL JOIN**

jointures

Mise en situation

Imaginons que nous sommes dans le contexte d'un site d'abonnements à un service qui propose à ses clients **trois types d'abonnements**:

- L'abonnement STANDARD
- L'abonnement PREMIUM
- L'abonnement VIP

Et nous avons d'autres part **les clients**. Comment rattacher les clients à leurs abonnements ?

Lier les tables: La clé étrangère (Foreign Key)

La clé étrangère

Avant de vouloir manipuler nos données, il est important de préparer nos tables à ce qu'elles communiquent entre elles. Ce moyen en SQL, c'est **la clé étrangère** (foreign key).
Elle se précise dès la construction de la table, il est donc préférable d'anticiper l'interaction entre les tables lorsque je construis une base de données.

Syntaxe

Pour avoir une jointure, il me faut d'abord deux tables, d'une part, **ma table Clients**:

```
CREATE TABLE Clients (
    id INT PRIMARY KEY AUTO_INCREMENT,
    first_name VARCHAR(100),
    last_name VARCHAR(100),
    city VARCHAR(100),
    age INT
  );
```

Syntaxe

D'autre part, **ma table Abonnements**, que je lie à ma table principale (ma table Clients ici):

```
CREATE TABLE Abonnements (
    client_id INT,
    abonnement_type VARCHAR(100),
    FOREIGN KEY (client_id)
    REFERENCES Clients(id)
);
```

Voilà, mes deux tables sont maintenant liées, c'est à dire que je ne pourrais plus insérer dans ma colonne client_id des valeurs qui n'existent pas déjà dans la colonne ID de ma table Client.

TOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

INNER JOIN

INNER JOIN

Il s'agit de ma première forme de jointure, en français on parle de **jointure intérieure**:


inner join

Son rôle est de connecter les lignes de deux tables (ou plus) en se basant sur une colonne qui les relie dans chacune.

Syntaxe

```
SELECT Clients.first_name, Clients.last_city, Abonnements.abonnement_type
FROM Clients
INNER JOIN Abonnements
ON Clients.id = Abonnements.client_id;
```

La clause `INNER JOIN` lie les deux tables Clients et Abonnements.
Le mot-clé `ON` est utilisé pour préciser la condition de jonction. Ici, c'est que l'id du client corresponde à celui du client_id de Abonnements.

Les ALIAS de tables

Les alias de tables

Lorsqu'on en vient à manipuler plusieurs tables, il devient fastidieux de préciser à chaque fois l'entièreté de la table avant chaque colonne. Mais l'**alias de table** nous permet, comme son nom l'indique, de pouvoir renommer notre table, exactement de la même manière qu'on le faisait pour les colonnes.

```
SELECT c.first_name, c.last_city, a.abonnement_type
FROM Clients AS c
INNER JOIN Abonnements AS a
ON c.id = a.client_id;
```

J'ai repris mon exemple précédent, on voit qu'il est beaucoup plus compacte de cette façon

Particularité: Joindre plus de deux tables

```
SELECT c.name, p.product, p.amount
FROM Clients AS c
JOIN Abonnements AS a
ON c.id = a.customer_id
JOIN Products AS p
ON a.product_id = p.product_id;
```

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

LEFT JOIN/RIGHT JOIN

LEFT JOIN/RIGHT JOIN

Une jointure gauche (ou son inverse la jointure droite) va récupérer l'entièreté de la table de gauche et va venir y ajouter les colonnes de la table de droite qui lui sont liées:


right join

On nomme left table: la table qui est mentionné la première dans la requête, la deuxième est la table de droite

Syntaxe

```
SELECT *
FROM Clients
LEFT JOIN Abonnements ON Clients.id = Abonnements.client_id;

SELECT *
FROM Clients
RIGHT JOIN Abonnements ON Clients.id = Abonnements.client_id;
```

Si aucune correspondance n'est trouvée dans la table de droite, les colonnes de la table de droite auront des valeurs NULL.

TOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

FULL JOIN

FULL JOIN

Enfin, la FULL JOIN est simplement la jointure totale de l'ensemble des deux tables


right join

ATTENTION: La clause FULL JOIN n'est pas prise en charge dans toutes les bases de données, notamment MySQL.

FULL JOIN en MySQL

En MySQL, on utilisera une combinaison de LEFT JOIN et RIGHT JOIN pour simuler un FULL JOIN grâce à la clause **UNION**.

```
SELECT *
FROM Clients
LEFT JOIN Abonnements ON Clients.id = Abonnements.client_id

UNION

SELECT *
FROM Clients
RIGHT JOIN Abonnements ON Clients.id = Abonnements.client_id
WHERE Clients.id IS NULL;
```

Exercice: Découverte des jointures

Considérez deux tables, "Clients" et "Achats".

Création des tables :

- La table "Clients" a les colonnes suivantes : `id` (clé primaire), `nom`, `prenom`, `ville`.
- La table "Achats" a les colonnes suivantes : `client_id` (clé étrangère vers "Clients"), `produit`, `montant`.

Insertion des données :

- Insérez au moins cinq clients dans la table "Clients".
- Insérez au moins dix achats dans la table "Achats", en veillant à ce que certains clients aient effectué des achats et d'autres non.

INNER JOIN :

- Sélectionnez les noms et prénoms des clients ainsi que les détails de leurs achats (si disponibles).

LEFT JOIN :

- Sélectionnez tous les clients et les détails de leurs achats s'ils ont effectué des achats, sinon affichez les colonnes des achats avec des valeurs NULL.

RIGHT JOIN :

- Sélectionnez tous les achats et les détails des clients correspondants, même s'il n'y a pas de correspondance.

FULL JOIN :

- Sélectionnez tous les clients et tous les achats, en affichant les détails des clients même s'ils n'ont pas effectué d'achats, et vice versa.

Correction :

a. INNER JOIN :

```
SELECT Clients.nom, Clients.prenom, Achats.produit, Achats.montant
FROM Clients
INNER JOIN Achats ON Clients.id = Achats.client_id;
```

b. LEFT JOIN :

```
SELECT Clients.nom, Clients.prenom, Achats.produit, Achats.montant
FROM Clients
LEFT JOIN Achats ON Clients.id = Achats.client_id;
```

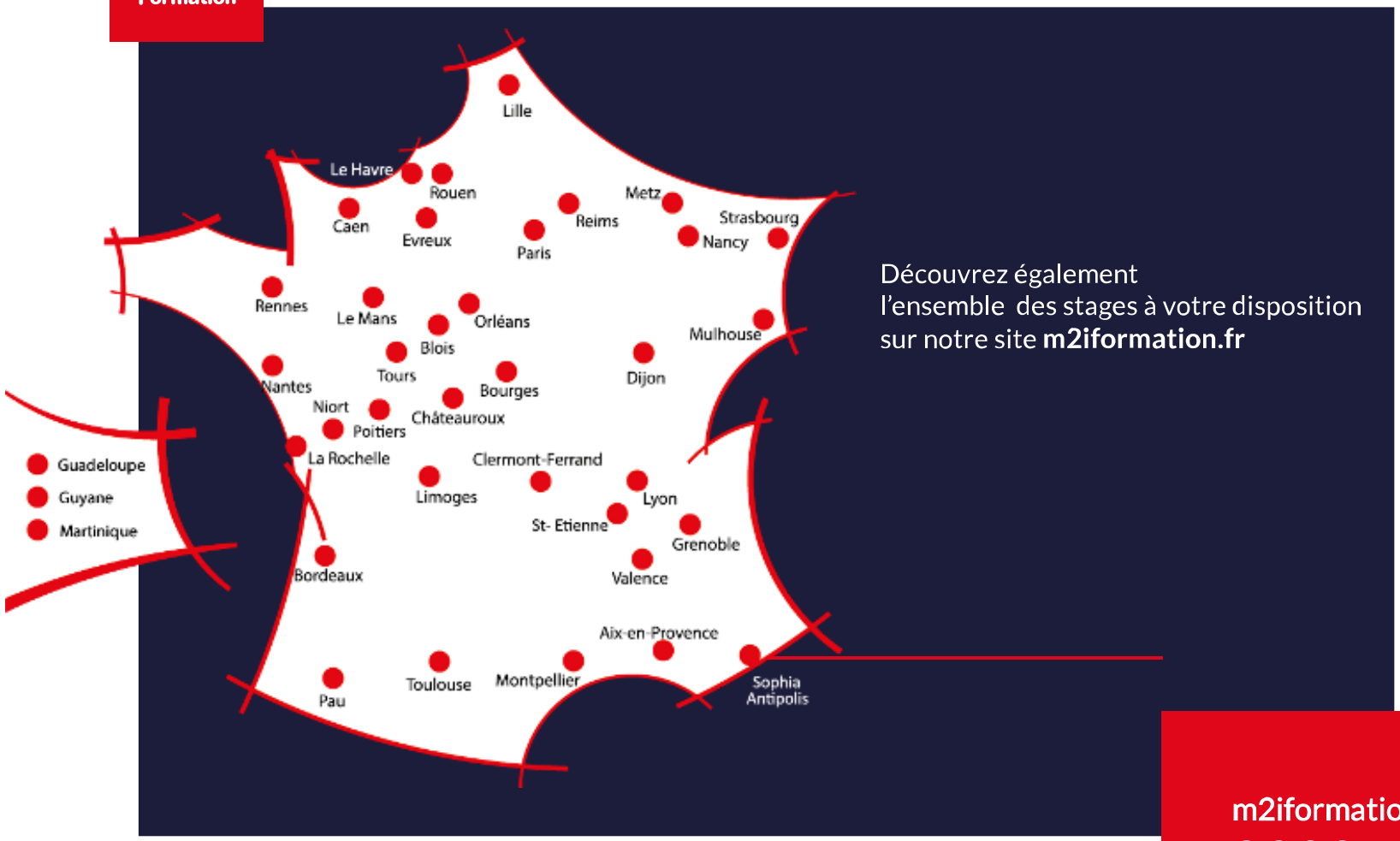
c. RIGHT JOIN :

```
SELECT Clients.nom, Clients.prenom, Achats.produit, Achats.montant
FROM Clients
RIGHT JOIN Achats ON Clients.id = Achats.client_id;
```

d. FULL JOIN :

```
SELECT Clients.nom, Clients.prenom, Achats.produit, Achats.montant
FROM Clients
FULL JOIN Achats ON Clients.id = Achats.client_id;
```

Comme vous pouvez le voir, la syntaxe de chacune des jointures est identique. Mais pas toujours le résultat de son affichage.



Découvrez également
l'ensemble des stages à votre disposition
sur notre site **m2iformation.fr**

m2iformation.fr