Les sous requêtes

420-3B3-SW Développement avec bases de données avancé

Requêtage des données

- Dans le langage SQL une sous-requête (aussi appelé « requête imbriquée » ou « requête en cascade ») consiste à exécuter une requête à l'intérieur d'une autre requête.
- Une requête imbriquée est souvent utilisée au sein d'une clause WHERE ou de HAVING pou remplacer une ou plusieurs constante.

Syntaxe

- Il y a plusieurs façons d'utiliser les sous-requêtes. De cette façon il y a plusieurs syntaxes envisageables pour utiliser des requêtes dans des requêtes:
  - Requête imbriquée qui retourne un seul résultat
  - Requête imbriquée qui retourne une colonne

Requêtage des données

Requête imbriquée qui retourne un seul résultat

- L'instruction SELECT peut être imbriquée :

1. Dans le WHERE

SELECT *liste_colonnes* **FROM** *nom_table(s)* **WHERE** *nom_colonne opérateur_comparaison** **(SELECT …)** ;

```
SELECT ProductName, Quantity FROM ProductDetails WHERE ProductId = (SELECT
ProductId FROM OrderDetails WHERE OrderId=100 LIMIT 1);
```

2. Dans le HAVING

SELECT *liste_colonnes* **FROM** *nom_table(s)* **WHERE** *condition*
HAVING *nom_colonne opérateur_comparaison** **(SELECT …)** ;

```
SELECT ProductId, Quantity FROM ProductDetails WHERE ProductCategory=2
HAVING ProductId = (SELECT ProductId FROM OrderDetails WHERE OrderId=100
LIMIT 1);
```

***A noter :** il est possible d'utiliser n'importe quel operateur de comparaison tel que =, >, <, >=, <=, <> ou LIKE.

Requêtage des données

Requête imbriquée qui retourne une colonne

- L'instruction SELECT peut être imbriquée :

1. Dans le WHERE

SELECT *liste_colonnes* **FROM** *nom_table(s)* **WHERE** *nom_colonne opérateur_comparaison** **(SELECT …)** ;

```
SELECT ProductName, Quantity FROM ProductDetails WHERE ProductId IN (SELECT
ProductId FROM OrderDetails WHERE OrderId=100);
```

2. Dans le HAVING

SELECT *liste_colonnes* **FROM** *nom_table(s)* **WHERE** *condition* **HAVING** *nom_colonne opérateur_comparaison** **(SELECT …)** ;

```
SELECT ProductId, Quantity FROM ProductDetails WHERE ProductCategory=2 HAVING
ProductId EXISTS (SELECT ProductId FROM OrderDetails WHERE OrderId=100 );
```

***A noter :** il est possible d'utiliser n'importe quel operateur de comparaison tel que IN, NOT IN, EXISTS ou NOT EXISTS…

Requêtage des données

Requête imbriquée qui retourne une colonne/un seul résultat

- L'instruction SELECT peut être imbriquée dans le WHERE ou HAVING avec d'autres opérateurs spéciaux comme:
  - ALL

SELECT *liste_colonnes* **FROM** *nom_table(s)* **WHERE** *nom_colonne opérateur_comparaison** **ALL** **(SELECT …)** ;

```
SELECT ProductName, Quantity FROM ProductDetails WHERE Quantity > ALL (SELECT Quantity FROM OrderDetails WHERE OrderId=100);
```

***A noter :** il est possible d'utiliser n'importe quel operateur de comparaison tel que =, >, <, >=, <=, <> ou LIKE.

Requêtage des données

Requête imbriquée qui retourne une colonne/un seul résultat

- L'instruction SELECT peut être imbriquée dans le WHERE ou HAVING avec d'autres opérateurs spéciaux comme:
  - ANY/SOME

SELECT *liste_colonnes* **FROM** *nom_table(s)* **WHERE** *nom_colonne opérateur_comparaison** **ANY/SOME** **(SELECT …)** ;

```
SELECT ProductName, Quantity FROM ProductDetails WHERE Quantity > ANY (SELECT Quantity FROM OrderDetails WHERE OrderId=100);
```

```
SELECT ProductName, Quantity FROM ProductDetails WHERE Quantity > SOME (SELECT Quantity FROM OrderDetails WHERE OrderId=100);
```

l'opérateur IN est équivalent de **= ANY**

***A noter :** il est possible d'utiliser n'importe quel operateur de comparaison tel que =, >, <, >=, <=, <> ou LIKE.

Requêtage des données

- L'instruction SELECT peut être imbriquée aussi dans un INSERT :

INSERT *nom_table* **SELECT** *liste_colonnes* **FROM** *nom_table(s)* **[WHERE** *condition(s)* **HAVING** *condition(s)* **ORDER BY** *nom_colonne type_ordre* **GROUP BY** *nom_colonne* **HAVING** *condition(s)***]** ;

```
INSERT Clothes SELECT ProductId, ProductName FROM ProductDetails WHERE ProductCategory=1;
```

```
INSERT ClothesAvailable SELECT ProductId, ProductName, Quantity FROM ProductDetails WHERE ProductCategory=1 HAVING Quantity > 0 ORDER BY ProductId DESC;
```

Les jointures

420-3B3-SW Développement avec bases de données avancé

Requêtage des données

- Les jointures en SQL permettent d'associer plusieurs tables dans une même requête. Cela permet d'exploiter la puissance des bases de données relationnelles pour obtenir des résultats qui combinent les données de plusieurs tables de manière efficace.
- Il y a plusieurs méthodes pour associer 2 tables ensemble. Voici la liste des principales techniques qui sont utilisées dans MYSQL:
  - INNER JOIN
  - CROSS JOIN
  - LEFT JOIN (ou LEFT OUTER JOIN)
  - RIGHT JOIN (ou RIGHT OUTER JOIN)
  - NATURAL JOIN

Requêtage des données

- INNER JOIN : jointure interne pour retourner les enregistrements quand la condition est vrai dans les 2 tables. C'est l'une des jointures les plus communes.
- CROSS JOIN : jointure croisée permettant de faire le produit cartésien de 2 tables. En d'autres mots, permet de joindre chaque ligne d'une table avec chaque ligne d'une seconde table. Attention, le nombre de résultats est en général très élevé.
- LEFT JOIN : jointure externe pour retourner tous les enregistrements de la table de gauche (LEFT = gauche) même si la condition n'est pas vérifié dans l'autre table.
- RIGHT JOIN : jointure externe pour retourner tous les enregistrements de la table de droite (RIGHT = droite) même si la condition n'est pas vérifié dans l'autre table.
- NATURAL JOIN : jointure naturelle entre 2 tables s'il y a au moins une colonne qui porte le même nom entre les 2 tables SQL.

Requêtage des données

INNER JOIN

Cette commande retourne les enregistrements lorsqu'il y a au moins une ligne dans chaque colonne qui correspond à la condition.

SELECT *liste_colonnes* **FROM** *nom_table1* **[*AS alias*]***
INNER JOIN *nom_table2* **[*AS alias*]* ON** *nom_table1.id = nom_table2.fk_id* ;

```
SELECT ProductId, ProductName FROM ProductDetails INNER JOIN OrderDetails ON
ProductDetails.ProductId = OrderDetails.ProductId ;
```

La syntaxe ci-dessus stipule qu'il faut sélectionner les enregistrements des tables table1 et table2 lorsque les données de la colonne « id » de table1 est égal aux données de la colonne fk_id de table2.

* **A noter** :l'utilisation de alias est facultative et au besoin

INNER JOIN

La jointure SQL peux aussi être écrite de la façon suivante :

SELECT *liste_colonnes* **FROM** *nom_table1* [***AS*** *alias*]*
INNER JOIN *nom_table2* [*AS alias*]* **WHERE** *nom_table1.id = nom_table2.fk_id* ;

```
SELECT ProductId, ProductName FROM ProductDetails INNER JOIN OrderDetails WHERE
ProductDetails.ProductId = OrderDetails.ProductId ;
```

La syntaxe avec la condition WHERE est une manière alternative de faire la jointure mais qui possède l'inconvénient d'être moins facile à lire s'il y a déjà plusieurs conditions dans le WHERE.

```
SELECT ProductId, ProductName FROM ProductDetails INNER JOIN OrderDetails WHERE
ProductDetails.ProductId = OrderDetails.ProductId AND ProductName LIKE 'A%';
```

* **A noter** :l'utilisation de alias est facultative et au besoin

Requêtage des données

Exemple

Imaginons une application qui possède une table utilisateur ainsi qu'une table commande qui contient toutes les commandes effectuées par les utilisateurs.

Table utilisateur :

id	prenom	nom	email	ville
1	Aimée	Marechal	aime.marechal@example.com	Paris
2	Esmée	Lefort	esmee.lefort@example.com	Lyon
3	Marine	Prevost	m.prevost@example.com	Lille
4	Luc	Rolland	lucrolland@example.com	Marseille

Table commande :

utilisateur_id	date_achat	num_facture	prix_total
1	2013-01-23	A00103	203.14
1	2013-02-14	A00104	124.00
2	2013-02-17	A00105	149.45
2	2013-02-21	A00106	235.35
5	2013-03-02	A00107	47.58

Requêtage des données

Exemple

Pour afficher toutes les commandes associées aux utilisateurs,
il est possible d'utiliser la requête suivante :

Table utilisateur :

id	prenom	nom	email	ville
1	Aimée	Marechal	aime.marechal@example.com	Paris
2	Esmée	Lefort	esmee.lefort@example.com	Lyon
3	Marine	Prevost	m.prevost@example.com	Lille
4	Luc	Rolland	lucrolland@example.com	Marseille

Table commande :

utilisateur_id	date_achat	num_facture	prix_total
1	2013-01-23	A00103	203.14
1	2013-02-14	A00104	124.00
2	2013-02-17	A00105	149.45
2	2013-02-21	A00106	235.35
5	2013-03-02	A00107	47.58

```
SELECT id, prenom, nom, date_achat, num_facture, prix_total FROM utilisateur INNER JOIN commande
ON utilisateur.id = commande.utilisateur_id ;
```

Résultats :

id	prenom	nom	date_achat	num_facture	prix_total
1	Aimée	Marechal	2013-01-23	A00103	203.14
1	Aimée	Marechal	2013-02-14	A00104	124.00
2	Esmée	Lefort	2013-02-17	A00105	149.45
2	Esmée	Lefort	2013-02-21	A00106	235.35

Requêtage des données

Exemple

Résultats :

id	prenom	nom	date_achat	num_facture	prix_total
1	Aimée	Marechal	2013-01-23	A00103	203.14
1	Aimée	Marechal	2013-02-14	A00104	124.00
2	Esmée	Lefort	2013-02-17	A00105	149.45
2	Esmée	Lefort	2013-02-21	A00106	235.35

Table utilisateur :

id	prenom	nom	email	ville
1	Aimée	Marechal	aime.marechal@example.com	Paris
2	Esmée	Lefort	esmee.lefort@example.com	Lyon
3	Marine	Prevost	m.prevost@example.com	Lille
4	Luc	Rolland	lucrolland@example.com	Marseille

Table commande :

utilisateur_id	date_achat	num_facture	prix_total
1	2013-01-23	A00103	203.14
1	2013-02-14	A00104	124.00
2	2013-02-17	A00105	149.45
2	2013-02-21	A00106	235.35
5	2013-03-02	A00107	47.58

- Le résultat de la requête montre parfaitement la jointure entre les 2 tables.
- Les utilisateurs 3 et 4 ne sont pas affichés puisqu'il n'y a pas de commandes associés à ces utilisateurs.

Attention : il est important de noter que si un utilisateur à été supprimé, alors on ne verra pas ses commandes dans la liste puisque INNER JOIN retourne uniquement les résultats ou la condition est vrai dans les 2 tables.

Requêtage des données

CROSS JOIN

Cette commande est un type de jointure sur 2 tables SQL qui permet de retourner le produit cartésien. Autrement dit, cela permet de retourner chaque ligne d'une table avec chaque ligne d'une autre table.

Exemple: effectuer le produit cartésien d'une table A qui contient 30 résultats avec une table B de 40 résultats va produire 1200 résultats (30 x 40 = 1200).

En général la commande CROSS JOIN est combinée avec la commande WHERE pour filtrer les résultats qui respectent certaines conditions.

Requêtage des données

CROSS JOIN

Attention, le nombre de résultat peut facilement être très élevé. S'il est effectué sur des tables avec beaucoup d'enregistrements, cela peut ralentir sensiblement le serveur.

Pour effectuer une jointure avec CROSS JOIN, il convient d'effectuer une requête SQL respectant la syntaxe suivante :

SELECT *liste_colonnes* **FROM** *nom_table1* **CROSS JOIN** *nom_table2* ;

```
SELECT MealName, DrinkName FROM Meals CROSS JOIN Drinks;
```

Méthode alternative pour retourner les mêmes résultats :

SELECT *liste_colonnes* **FROM** *nom_table1*, *nom_table2* ;

```
SELECT MealName, DrinkName FROM Meals, Drinks;
```

Requêtage des données

CROSS JOIN

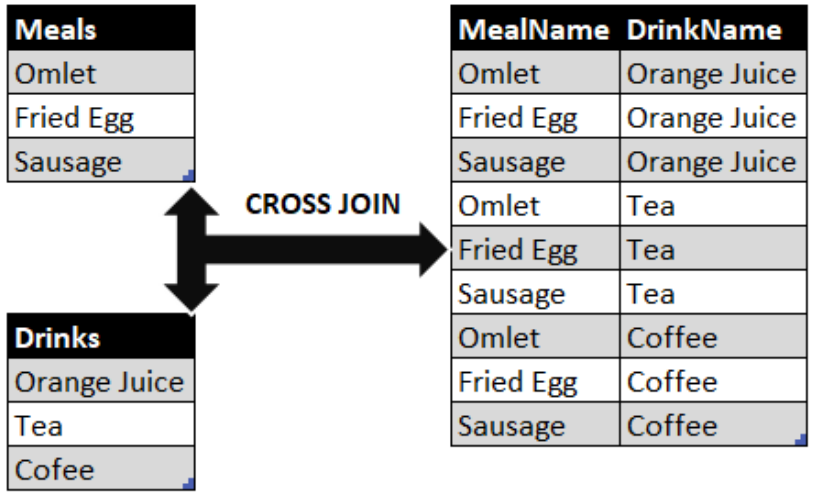
SELECT *liste_colonnes* **FROM** *nom_table1* **CROSS JOIN** *nom_table2* ;

```
SELECT MealName, DrinkName FROM Meals CROSS JOIN Drinks;
```

Méthode alternative pour retourner les mêmes résultats :

SELECT *liste_colonnes* **FROM** *nom_table1*, *nom_table2* ;

```
SELECT MealName, DrinkName FROM Drinks,Meals;
```



Requêtage des données

Exemple

Imaginons une application de recettes de cuisines qui contient 2 tables d'ingrédients, la table légume et la table fruit.

Table legume :

l_id	l_nom_fr_fr	l_nom_en_gb
45	Carotte	Carott
46	Oignon	Onion
47	Poireau	Leek

Table fruit :

f_id	f_nom_fr_fr	f_nom_en_gb
87	Banane	Banana
88	Kiwi	Kiwi
89	Poire	Pear

Requêtage des données

Exemple

Pour une raison quelconque l'application doit associer tous les légumes avec tous les fruits. Toutes les combinaisons doivent être affichées. Pour cela il convient d'effectuer l'une ou l'autre des requêtes suivantes :

```
SELECT l_id, l_nom_fr_fr, f_id, f_nom_fr_fr FROM legume CROSS JOIN fruit;
```

ou :

```
SELECT l_id, l_nom_fr_fr, f_id, f_nom_fr_fr FROM legume, fruit;
```

Table legume :

l_id	l_nom_fr_fr	l_nom_en_gb
45	Carotte	Carott
46	Oignon	Onion
47	Poireau	Leek

Table fruit :

f_id	f_nom_fr_fr	f_nom_en_gb
87	Banane	Banana
88	Kiwi	Kiwi
89	Poire	Pear

Requêtage des données

Exemple

```
SELECT l_id, l_nom_fr_fr, f_id, f_nom_fr_fr FROM legume CROSS JOIN fruit;
```

ou :

```
SELECT l_id, l_nom_fr_fr, f_id, f_nom_fr_fr FROM legume, fruit;
```

Résultats :

l_id	l_nom_fr_fr	f_id	f_nom_fr_fr
45	Carotte	87	Banane
45	Carotte	88	Kiwi
45	Carotte	89	Poire
46	Oignon	87	Banane
46	Oignon	88	Kiwi
46	Oignon	89	Poire
47	Poireau	87	Banane
47	Poireau	88	Kiwi
47	Poireau	89	Poire

Le résultat montre bien que chaque légume est associé à chaque fruit.
Avec 3 fruits et 3 légumes, il y a donc 9 lignes de résultats (3 x3 = 9).

Requêtage des données

LEFT JOIN

Cette commande permet de lister tous les résultats de la table de gauche (left = gauche) même s'il n'y a pas de correspondance dans la deuxième table.

Pour lister les enregistrement de table1, même s'il n'y a pas de correspondance avec table2, il convient d'effectuer une requête SQL utilisant la syntaxe suivante:

SELECT *liste_colonnes* **FROM** *nom_table1* **LEFT [OUTER] JOIN** *nom_table2* **ON** *nom_table1.id = nom_table2.fk_id*;

```
SELECT ProductId, ProductName, OrderId FROM ProductDetails LEFT JOIN
OrderDetails ON ProductDetails.ProductId = OrderDetails.ProductId ;
```

Requêtage des données

LEFT JOIN

Cette requête est particulièrement intéressante pour récupérer les informations de table1 tout en récupérant les données associées, même s'il n'y a pas de correspondance avec table2. A savoir, s'il n'y a pas de correspondance les colonnes de table2 vaudront toutes NULL.

Requêtage des données

Exemple

Imaginons une application contenant des utilisateurs et des commandes pour chacun de ces utilisateurs. La base de données de cette application contient une table pour les utilisateurs et sauvegarde leurs achats dans une seconde table. Les 2 tables sont reliées grâce à la colonne *utilisateur_id* de la table des commandes. Cela permet d'associer une commande à un utilisateur.

Requêtage des données

Exemple

Table utilisateur :

id	prenom	nom	email	ville
1	Aimée	Marechal	aime.marechal@example.com	Paris
2	Esmée	Lefort	esmee.lefort@example.com	Lyon
3	Marine	Prevost	m.prevost@example.com	Lille
4	Luc	Rolland	lucrolland@example.com	Marseille

Table commande :

utilisateur_id	date_achat	num_facture	prix_total
1	2013-01-23	A00103	203.14
1	2013-02-14	A00104	124.00
2	2013-02-17	A00105	149.45
2	2013-02-21	A00106	235.35
5	2013-03-02	A00107	47.58

Requêtage des données

Exemple

Pour lister tous les utilisateurs avec leurs commandes et afficher également les utilisateurs qui n'ont pas effectuées d'achats, il est possible d'utiliser la requête suivante :

```
SELECT id,prenom,nom,date_achat, num_facture, prix_total FROM utilisateur LEFT JOIN commande ON utilisateur.id = commande.utilisateur_id ;
```

Résultats :

id	prenom	nom	date_achat	num_facture	prix_total
1	Aimée	Marechal	2013-01-23	A00103	203.14
1	Aimée	Marechal	2013-02-14	A00104	124.00
2	Esmée	Lefort	2013-02-17	A00105	149.45
2	Esmée	Lefort	2013-02-21	A00106	235.35
3	Marine	Prevost	NULL	NULL	NULL
4	Luc	Rolland	NULL	NULL	NULL

Les dernières lignes montrent des utilisateurs qui n'ont effectué aucune commande. La ligne retourne la valeur NULL pour les colonnes concernant les achats qu'ils n'ont pas effectués.

Requêtage des données

Filtrer sur la valeur NULL

Attention, la valeur NULL n'est pas une chaîne de caractère. Pour filtrer sur ces caractères il faut utiliser la commande IS NULL.

Par exemple, pour lister les utilisateurs qui n'ont pas effectués d'achats il est possible d'utiliser la requête suivante:

```
SELECT id, prenom, nom, utilisateur_id FROM utilisateur LEFT JOIN commande ON
utilisateur.id = commande.utilisateur_id WHERE utilisateur_id IS NULL;
```

Résultats :

id	prenom	nom	utilisateur_id
3	Marine	Prevost	NULL
4	Luc	Rolland	NULL

RIGHT JOIN

la commande RIGHT JOIN (ou RIGHT OUTER JOIN) est un type de jointure entre 2 tables qui permet de retourner tous les enregistrements de la table de droite (right = droite) même s'il n'y a pas de correspondance avec la table de gauche.

S'il y a un enregistrement de la table de droite qui ne trouve pas de correspondance dans la table de gauche, alors les colonnes de la table de gauche auront NULL pour valeur.

SELECT *liste_colonnes* **FROM** *nom_table1* **RIGHT [OUTER] JOIN** *nom_table2* **ON** *nom_table1.id = nom_table2.fk_id*;

```
SELECT ProductId, ProductName, OrderId FROM ProductDetails RIGHT JOIN OrderDetails ON ProductDetails.ProductId = OrderDetails.ProductId ;
```

RIGHT JOIN

Cette syntaxe stipule qu'il faut lister toutes les lignes du tableau table2 (tableau de droite) et afficher les données associées du tableau table1 s'il y a une correspondance entre ID de table1 et FK_ID de table2.

S'il n'y a pas de correspondance, l'enregistrement de table2 sera affiché et les colonnes de table1 vaudront toutes NULL.

Requêtage des données

Exemple

Prenons l'exemple d'une base de données qui contient des utilisateurs et un historique d'achat de ces utilisateurs. Cette 2 tables sont reliées entre grâce à la colonne *utilisateur_id* de la table des commandes. Cela permet de savoir à quel utilisateur est associé un achat.

Table utilisateur :

id	prenom	nom	email	ville	actif
1	Aimée	Marechal	aime.marechal@example.com	Paris	1
2	Esmée	Lefort	esmee.lefort@example.com	Lyon	0
3	Marine	Prevost	m.prevost@example.com	Lille	1
4	Luc	Rolland	lucrolland@example.com	Marseille	1

Table commande :

utilisateur_id	date_achat	num_facture	prix_total
1	2013-01-23	A00103	203.14
1	2013-02-14	A00104	124.00
2	2013-02-17	A00105	149.45
3	2013-02-21	A00106	235.35
5	2013-03-02	A00107	47.58

Requêtage des données

Exemple

Pour afficher toutes les commandes avec le nom de l'utilisateur correspondant il est normal d'habitude d'utiliser INNER JOIN en SQL. Malheureusement, si l'utilisateur a été supprimé de la table, alors ça ne retourne pas l'achat.

L'utilisation de RIGHT JOIN permet de retourner tous les achats et d'afficher le nom de l'utilisateur s'il existe. Pour cela il convient d'utiliser cette requête :

```
SELECT id, prenom, nom, utilisateur_id, date_achat, num_facture FROM utilisateur RIGHT JOIN commande ON utilisateur.id = commande.utilisateur_id;
```

Requêtage des données

Exemple

```
SELECT id, prenom, nom, utilisateur_id, date_achat, num_facture FROM utilisateur RIGHT JOIN commande ON utilisateur.id = commande.utilisateur_id;
```

Résultats :

id	prenom	nom	utilisateur_id	date_achat	num_facture
1	Aimée	Marechal	1	2013-01-23	A00103
1	Aimée	Marechal	1	2013-02-14	A00104
2	Esmée	Lefort	2	2013-02-17	A00105
3	Marine	Prevost	3	2013-02-21	A00106
NULL	NULL	NULL	5	2013-03-02	A00107

Ce résultat montre que la facture A00107 est liée à l'utilisateur numéro 5. Or, cet utilisateur n'existe pas ou n'existe plus.

Grâce à RIGHT JOIN, l'achat est tout de même affiché mais les informations liées à l'utilisateur sont remplacées par NULL.

NATURAL JOIN

Cette jointure s'effectue à la condition qu'il y ai des colonnes du même nom et de même type dans les 2 tables. Le résultat d'une jointure naturelle est la création d'un tableau avec autant de lignes qu'il y a de paires correspondant à l'association des colonnes de même nom.

La jointure naturelle de 2 tables peut s'effectuer facilement, comme le montre la requête SQL suivante :

```
SELECT liste_colonnes FROM nom_table1  NATURAL JOIN nom_table2 ;
```

L'avantage d'un NATURAL JOIN c'est qu'il n'a pas besoin d'utiliser la clause ON.

Requêtage des données

Exemple

Une utilisation classique d'une telle jointure pourrait être l'utilisation dans une application qui utilise une table utilisateur et une table pays. Si la table utilisateur contient une colonne pour l'identifiant du pays, il sera possible d'effectuer une jointure naturelle.

Table « utilisateur » :

user_id	user_prenom	user_ville	pays_id
1	Jérémie	Paris	1
2	Damien	Lyon	2
3	Sophie	Marseille	NULL
4	Yann	Lille	9999
5	Léa	Paris	1

Table « pays » :

pays_id	pays_nom
1	France
2	Canada
3	Belgique
4	Suisse

Requêtage des données

Exemple

Pour avoir la liste de tous les utilisateurs avec le pays correspondant, il est possible d'effectuer une requête SQL similaire à celle-ci :

```
SELECT * FROM utilisateur NATURAL JOIN pays;
```

Cette requête retournera le résultat suivant :

pays_id	user_id	user_prenom	user_ville	pays_nom
1	1	Jérémie	Paris	France
2	2	Damien	Lyon	Canada
NULL	3	Sophie	Marseille	NULL
9999	4	Yann	Lille	NULL
1	5	Léa	Paris	France

Cet exemple montre qu'il y a bien eu une jointure entre les 2 tables grâce à la colonne « *pays_id* » qui se trouve dans l'une et l'autre des tables.