T. D. n° 8
Jointure en MySQL

Résumé

Ce document est le TD n° 8 du module **Base de données**. Il reprend rapidement des éléments du cours et propose une mise en pratique interactive des jointures.

1 Jointure

Reprenez votre base de donnée. Cette base contient *a minima* les tables suivantes :

```
mysql> describe Animal;
+----------------+----------------------+------+-----+---------+----------------+
| Field          | Type                 | Null | Key | Default |
   Extra          |
+----------------+----------------------+------+-----+---------+----------------+
| id             | smallint(5) unsigned | NO   | PRI | NULL    |
   auto_increment |
| espece         | varchar(40)          | NO   | MUL | NULL    |
                  |
| sexe           | char(1)              | YES  |     | NULL    |
                  |
| date_naissance | datetime             | NO   |     | NULL    |
                  |
| nom            | varchar(30)          | YES  | MUL | NULL    |
                  |
| commentaires   | text                 | YES  |     | NULL    |
                  |
```

```
| espece_id      | smallint(5) unsigned | NO   | MUL | NULL    |
                  |
| race_id        | smallint(5) unsigned | YES  | MUL | NULL    |
                  |
| mere_id        | smallint(5) unsigned | YES  | MUL | NULL    |
                  |
| pere_id        | smallint(5) unsigned | YES  | MUL | NULL    |
                  |
+-------------+----------------------+------+-----+---------+----------------+
```

```
mysql> describe Espece;
+-------------+----------------------+------+-----+---------+----------------+

| Field       | Type                 | Null | Key | Default | Extra
             |
+-------------+----------------------+------+-----+---------+----------------+

| id          | smallint(5) unsigned | NO   | PRI | NULL    |
   auto_increment |
| nom_courant | varchar(40)          | NO   |     | NULL    |
                  |
| nom_latin   | varchar(40)          | NO   | UNI | NULL    |
                  |
| description | text                 | YES  |     | NULL    |
                  |
+-------------+----------------------+------+-----+---------+----------------+
```

```
mysql> describe Race;
+-------------+----------------------+------+-----+---------+----------------+

| Field       | Type                 | Null | Key | Default | Extra
             |
+-------------+----------------------+------+-----+---------+----------------+

| id          | smallint(5) unsigned | NO   | PRI | NULL    |
   auto_increment |
| nom         | varchar(40)          | NO   |     | NULL    |
                  |
| espece_id   | smallint(5) unsigned | NO   | MUL | NULL    |
                  |
| description | text                 | YES  |     | NULL    |
                  |
+-------------+----------------------+------+-----+---------+----------------+
```

Les jointures vous permettent d'interroger plusieurs tables dans la même requête. Le principe des jointures est de joindre plusieurs tables. Pour ce faire, on utilise les informations communes des tables (notamment les clés étrangères).
Lorsque vous avez ajouté dans votre base les informations sur les espèces (leur nom latin et leur description), vous avez constaté que c'était une très mauvaise idée

de tout mettre dans la table Animal, car il nous faudrait alors garder la même nomenclature pour tous les chiens, la même pour toutes les tortues, etc. Cependant, vous avez sans doute remarqué que, si vous voulez afficher la description de l'espèce de Cartouche (votre petit préféré), vous avez besoin de deux requêtes :

— on trouve l'id de l'espèce de Cartouche grâce à la table Animal.
— on trouve la description de l'espèce grâce à son id.

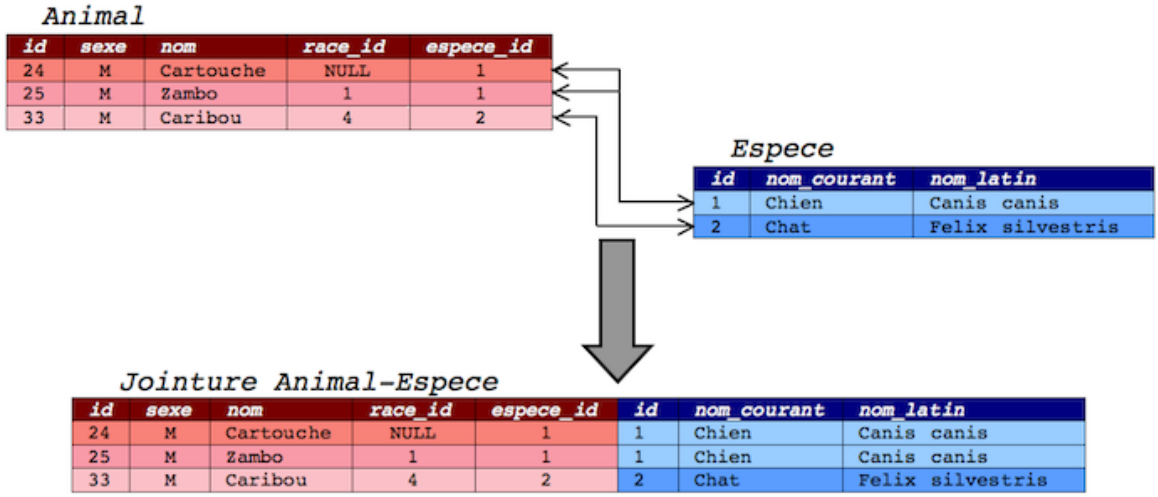
Vous devez donc envoyer les requêtes suivantes :

```
SELECT espece_id FROM Animal WHERE nom = 'Cartouche';
SELECT description FROM Espece WHERE id = 1;
```

Plutôt que de faire deux requêtes successives, il est possible d'obtenir ce résultat via une seule requête. C'est là que les jointures entrent en jeu ; on va utiliser l'information commune entre les deux tables : l'id de l'espèce, qui est présente dans Animal (la colonne espece_id), et l'id dans la table Espece (la colonne id).

```
SELECT Espece.description

FROM Espece

INNER JOIN Animal

    ON Espece.id = Animal.espece_id

WHERE Animal.nom = 'Cartouche';
```

En fait, lorsque l'on fait une jointure, on crée une table virtuelle et temporaire qui reprend les colonnes des tables liées. Le schéma ci-dessous illustre ce principe.
Au départ, on a deux tables : Animal(id, sexe, nom, race_id, espece_id) et Espece(id, nom_courant, nom_latin). Les deux premières lignes d'Animal correspondent à la première ligne d'Espece, et la troisième ligne d'Animal à la deuxième ligne d'Espece. Une fois les deux tables jointes, on obtient une table possédant toutes les colonnes d'Animal et toutes les colonnes d'Espece, avec les valeurs correspondantes de chaque table. On peut voir que les cinquième et sixième colonnes de la table de jointure ont les mêmes valeurs. Ensuite, de cette table virtuelle, on peut extraire ce que l'on veut. La colonne nom_latin pour la ligne ayant "Caribou" dans la colonne nom, par exemple.



Voici donc la syntaxe à utiliser pour faire des requêtes avec jointure(s) interne(s).

```
SELECT *                                        -- comme d'habitude,
   vous sélectionnez les colonnes que vous voulez

FROM nom_table1

[INNER] JOIN nom_table2                         -- INNER explicite le
   fait qu'il s'agit d'une jointure interne, mais c'est facultatif

    ON colonne_table1 = colonne_table2          -- sur quelles colonnes
       se fait la jointure

                                                -- vous pouvez mettre
                                                   colonne_table2 =
                                                   colonne_table1, l'
                                                   ordre n'a pas d'
                                                   importance


[WHERE ...]

[ORDER BY ...]                                  -- les clauses
   habituelles sont bien sûr utilisables !

[LIMIT ...]
```

La clause `ON` sert à préciser la condition de la jointure. C'est-à-dire sur quel(s) critère(s) les deux tables doivent être jointes. Dans la plupart des cas, il s'agira d'une condition d'égalité simple, comme `ON` Animal.espece_id = Especе.id. Il est cependant tout à fait possible d'avoir plusieurs conditions à remplir pour lier les deux tables. On utilise alors les opérateurs logiques habituels. Par exemple, une jointure peut très bien se faire sur plusieurs colonnes :

```
SELECT *

FROM table1
```

```
INNER JOIN table2

   ON table1.colonneA = table2.colonneJ

      AND table1.colonneT = table2.colonneX

      [AND ...];
```

Il peut arriver que vous ayez dans vos deux tables des colonnes portant le même nom. C'est le cas dans notre exemple, puisque la table Animal comporte une colonne id, tout comme la table Espece. Il est donc important de préciser de quelle colonne on parle dans ce cas-là. Vous l'avez vu dans notre requête, on utilise pour cela l'opérateur '.' (nom_table.nom_colonne). Pour les colonnes ayant un nom non ambigu (qui n'existe dans aucune autre table de la jointure), il n'est pas obligatoire de préciser la table. En général, il est conseillé de préciser la table quand il s'agit de grosses requêtes avec plusieurs jointures. En revanche, pour les petites jointures courantes, il est vrai que c'est moins long à écrire si l'on ne précise pas la table.

À vous !

a) sélectionnez avec `INNER JOIN` le nom des animaux commençant par "Ch", ainsi que de l'id et la description de leur espèce.

b) sélectionnez avec `INNER JOIN` le nom des animaux née avant 2008-03-15 12 :02 :00, ainsi que l'id, le nom courant et la description de leur espèce.

c) essayez de sélectionner avec `INNER JOIN` le nom des animaux née avant 2008-03-15 12 :02 :00, ainsi que, le nom courant, la description de leur espèce mais pas leurs id.

d) essayez la même séléction mais sans le `INNER`. Que pouvez vous conclure sur les contraintes des `INNER JOIN` ?

e) Sélectionnez des nom d'animaux dont les espèces ont un t dans la description.

2 Les alias

Pour réaliser des jointure, MYSQL utilise des alias. Les alias sont des noms de remplacement que l'on donne de manière temporaire (le temps d'une requête) à une colonne, une table, une donnée. Les alias sont introduits par le mot-clé `AS`. Ce mot-clé est facultatif, vous pouvez très bien définir un alias sans utiliser `AS`.
Les alias sont souvent utilisés avec les jointures. Ils permettent notamment de renommer les tables dans des variables temporaires, et ainsi d'écrire moins de code.
Par exemple : on renomme la table Espece "e", et la table Animal "a".

```
SELECT e.id,

       e.description,

       a.nom
```

```
FROM Espece AS e          -- On donne l'alias "e" à Espece

INNER JOIN Animal AS a    -- et l'alias "a" à Animal.

     ON e.id = a.espece_id

WHERE a.nom LIKE 'Ch%';
```

Comme vous le voyez, le code est plus compact. Ici encore, c'est souvent untilisé pour de petites requêtes ponctuelles. Par contre, pour de grosses requêtes, il est conseillé d'utiliser des noms explicites ; c'est ainsi plus facile de s'y retrouver.

Une autre utilité des alias est de renommer les colonnes pour que le résultat soit plus clair. Observez le résultat de la requête précédente. Vous avez trois colonnes : id, description et nom. Le nom de la table dont provient la colonne n'est indiqué nulle part. A priori, vous savez ce que vous avez demandé. Il n'y a pas encore trop de colonnes, mais imaginez que vous sélectionniez une vingtaine de colonnes. Ce serait quand même mieux de savoir de quel id on parle, s'il s'agit du nom de l'animal, de son maître, du père, du fils ou du Saint-Esprit ! Il est intéressant là aussi d'utiliser les alias.

À vous !

a) En utilisant un alias pour chaque table, sélectionnez avec `JOIN` le nom des animaux née avant 2008-03-15 12 :02 :00, ainsi que l'id, le nom courant et la description de leur espèce.

b) En utilisant un alias pour la table Animal, sélectionnez des nom d'animaux dont les espèces ont un t dans la description.

3 INNER JOIN

`INNER JOIN` permet de faire une jointure interne sur deux tables. Mais que signifie donc ce "interne" ?
C'est très simple ! Lorsque l'on fait une jointure interne, cela veut dire que l'on exige qu'il y ait des données de part et d'autre de la jointure. Donc, si l'on fait une jointure sur la colonne a de la table A et la colonne b de la table B :

```
SELECT *
FROM A
INNER JOIN B
    ON A.a = B.b
```

Ceci retournera uniquement les lignes pour lesquelles A.a et B.b correspondent.

À vous !

a) Formulez une requête permettant de voir toutes les races des chats de la table animaux via un `INNER JOIN`

b) Cherchez la *race_id* de Choupi et Roucky. Apparaissent il dans la requête précédente ? Pourquoi ?

c) Pourquoi la race "Sphynx" n'apparaît pas dans le résultat de la première requête ?

4 Jointure externe

Comme je viens de vous le dire, une jointure externe permet de sélectionner également les lignes pour lesquelles il n'y a pas de correspondance dans une des tables jointes. MySQL permet deux types de jointures externes : les jointures par la gauche et les jointures par la droite.

4.1 Jointures par la gauche

Lorsque l'on fait une jointure par la gauche (grâce aux mots-clés `LEFT JOIN` ou `LEFT OUTER JOIN`), cela signifie que l'on veut toutes les lignes de la table de gauche (sauf restrictions dans une clause `WHERE`, bien sûr), même si certaines n'ont pas de correspondance avec une ligne de la table de droite. Alors, table de gauche, table de droite, laquelle est laquelle ? C'est très simple, nous lisons de gauche à droite, donc la table de gauche est la première table mentionnée dans la requête, c'est-à-dire, en général, la table donnée dans la clause `FROM`.

```
SELECT *
LEFT A
LEFT JOIN B
    ON A.a = B.b
```

Ceci retournera les lignes associées à A.a et leurs correspondances éventuelles avec B.b.

4.2 Jointures par la droite

Les jointures par la droite (`RIGHT JOIN` ou `RIGHT OUTER JOIN`), c'est évidemment le même principe, sauf que ce sont toutes les lignes de la table de droite qui sont sélectionnées, même s'il n'y a pas de correspondance dans la table de gauche.

À vous !

a) Générer la requêtes pour connaître la race des chats, mais que cette fois-ci il faut également afficher les chats qui n'ont pas de race (via un `LEFT JOIN`). Utilisez un alias nom_animal et un alias race pour renommer les colonnes affichés.

b) Générer la requêtes pour connaître la race des chats, mais que cette fois-ci il faut également afficher les chats qui n'ont pas de race (via un `RIGHT JOIN`).

c) Générer la requêtes pour connaître la race des chats (en affichant également les chats qui n'ont pas de race), dont le nom commence par 'C'.(via un `LEFT JOIN`).

d) Générer la requêtes pour connaître la race des chats (en affichant également les chats qui n'ont pas de race), dont le nom commence par 'C'.(via un `RIGHT JOIN`).

e) Vérifiez que Choupi apparaît dans votre résultat.

f) Générer la requêtes pour connaître les races chats, mais que cette fois-ci il faut également afficher les races qui n'ont pas de chats dans la table Animal (via un `LEFT JOIN` puis un `RIGHT JOIN`).

5 Jointure avec plusieurs tables

Il est possible de relier plus de deux tables entre elles. Pour cela, vous pouvez enchaîner les `JOIN` à la suite. Par exemple

```
SELECT *
FROM T1
INNER JOIN T2
ON T1.col_a = T2.col_b
INNER JOIN T3
ON T1.col_c = T3.col_d
```

À vous !

a) Vous devez obtenir la liste des races de chiens qui sont des chiens de berger. (On considère, même si ce n'est pas tout à fait vrai, que les chiens de berger ont "berger" dans leur nom de race.)

b) Vous devez obtenir la liste des animaux (leur nom, date de naissance et race) pour lesquels nous n'avons aucune information sur leur pelage. Dans la description des races, j'utilise parfois "pelage", parfois "poil", et parfois "robe".

c) Vous devez obtenir la liste des chats et des perroquets amazones, avec leur sexe, leur espèce (nom latin) et leur race s'ils en ont une. Regroupez les chats ensemble, les perroquets ensemble et, au sein de l'espèce, regroupez les races.

d) Vous devez obtenir la liste des chiennes dont on connaît la race, et qui sont en âge de procréer (c'est-à-dire nées avant juillet 2016). Affichez leurs nom, date de naissance et race.

e) Vous devez obtenir la liste des chats dont on connaît les parents, ainsi que le nom de ces parents. (Alias obligatoire)

f) Vous devez maintenant obtenir la liste des enfants de Bilba (nom, sexe et date de naissance).

g) Vous devez obtenir la liste des animaux dont on connaît le père, la mère, la race, la race du père, la race de la mère. Affichez le nom et la race de l'animal et de ses parents, ainsi que l'espèce de l'animal (pas des parents).