Jointures en SQL, quelques notes

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 6 mai 2009

https://www.bortzmeyer.org/jointures-sql.html

Je ne vais pas faire le Nième tutoriel sur les jointures SQL (pour cela, je recommande l'excellent article du Wikipédia anglophone), j'essaie juste de documenter deux ou trois choses de base sur elles, pour m'en souvenir la prochaine fois. Si ça peut servir à d'autres...

Une jointure, c'est... joindre deux tables, en général avec un critère de sélection. L'idée de base est de générer des tuples qui combinent les tuples des deux bases. Ce n'est donc **pas** une opération ensembliste, contrairement à ce qu'à écrit Jeff Atwood <http://www.codinghorror.com/blog/archives/000976.html>.

Avant de commencer, je crée quelques tables pour illustrer la question, une table de personnes et une de livres (le code SQL complet, testé avec PostgreSQL est disponible en ligne (en ligne sur https://www.bortzmeyer.org/files/create-for-joins.sql)) :

```
CREATE TABLE Authors (
   id SERIAL UNIQUE NOT NULL,
   name TEXT NOT NULL);

CREATE TABLE Books (
   id SERIAL UNIQUE NOT NULL,
   title TEXT NOT NULL,
   author INTEGER REFERENCES Authors(id));
```

Bien sûr, ce schéma de données est ultra-simplifié. Par exemple, il ne permet pas de représenter un livre qui a plusieurs auteurs. Mais peu importe, c'est pour faire des exemples. Le point important à noter est qu'un « auteur » n'est pas forcément référencé par un livre et qu'un livre peut ne pas avoir d'auteur connu (la colonne `author` peut être `NULL`).

Voici les données, avec des livres que j'ai apprécie (comme « La horde du contrevent <https://www.bortzmeyer.org/horde-du-contrevent.html> » ou les livres de Fred Vargas) :

```
essais=> SELECT * FROM Authors;
 id |        name
----+--------------------
  1 | Stéphane Bortzmeyer
  2 | Fred Vargas
  3 | Ève Demazière
  4 | Alain Damasio
(4 rows)

essais=> SELECT * FROM Books;
 id |                  title                  | author
----+-----------------------------------------+--------
  1 | Les cultures noires d'Amérique Centrale |      3
  2 | La horde du contrevent                  |      4
  3 | Pars vite et reviens tard               |      2
  4 | L'homme à l'envers                      |      2
  5 | Bible                                   |
(5 rows)
```

(Cherchez le livre sans auteur connu et la personne qui n'a pas écrit de livre.)

Il existe deux grands types de jointures, la jointure interne (`INNER JOIN`) et l'externe (`OUTER JOIN`). S'il existe toujours un lien entre les deux tables (si tous les livres ont un auteur et si tous les auteurs sont référencés depuis au moins un livre), il n'existe pas de différence entre les deux types. Mais ce n'est pas le cas en général.

La jointure interne s'écrit avec le mot-clé `INNER JOIN` et la condition `ON` (l'ensemble des requêtes de jointure est disponible en ligne (en ligne sur `https://www.bortzmeyer.org/files/joins.sql`), tous les tests ont été faits avec PostgreSQL 8.3) :

```
essais=> SELECT name, title FROM Books INNER JOIN Authors
                ON Books.author = Authors.id;
     name      |                  title
---------------+-----------------------------------------
 Ève Demazière | Les cultures noires d'Amérique Centrale
 Alain Damasio | La horde du contrevent
 Fred Vargas   | Pars vite et reviens tard
 Fred Vargas   | L'homme à l'envers
(4 rows)
```

(Le `INNER` est facultatif, la jointure interne est la jointure par défaut.) Si les deux colonnes sur lesquelles se fait la jointure ont le même nom, on peut utiliser `USING` au lieu de `ON`.

Les jointures internes s'écrivent plus fréquemment (mais c'est une question de goût `<http://stackoverflow.com/questions/334201/why-isnt-sql-ansi-92-standard-better-adopted-over-` avec l'ancienne syntaxe :

```
essais=> SELECT name, title FROM Books, Authors
                 WHERE Books.author = Authors.id;
     name      |                  title
---------------+-----------------------------------------
 Ève Demazière | Les cultures noires d'Amérique Centrale
 Alain Damasio | La horde du contrevent
 Fred Vargas   | Pars vite et reviens tard
 Fred Vargas   | L'homme à l'envers
(4 rows)
```

Dans les deux cas (c'était juste une différence de syntaxe), le résultat est le même et n'inclus pas les livres sans auteur connu, ni les « auteurs » qui n'ont pas écrit de livre. Une jointure interne ne sélectionne que les tuples qui répondent à la condition de jointure (le `ON`, dans la nouvelle syntaxe).

Et les jointures externes ? Au contraire des internes, elles produisent également des tuples où la condition de jointure n'est pas remplie. Un mot clé `LEFT` ou `RIGHT` est obligatoire. Une jointure gauche garde les tuples de la table indiquée à gauche et vous vous doutez de ce que fera une jointure droite. On peut facilement transformer l'une en l'autre, `A LEFT OUTER JOIN B` est la même chose que `B RIGHT OUTER JOIN A`. Je ne montrerai donc que les jointures externes à gauche :

```
essais=> SELECT name, title FROM Books LEFT OUTER JOIN Authors
             ON Books.author = Authors.id;
     name      |                 title
---------------+----------------------------------------
 Ève Demazière | Les cultures noires d'Amérique Centrale
 Alain Damasio | La horde du contrevent
 Fred Vargas   | Pars vite et reviens tard
 Fred Vargas   | L'homme à l'envers
               | Bible
(5 rows)
```

On voit désormais le livre sans auteur. Si la colonne `name` vide gène :

```
essais=> SELECT CASE WHEN name IS NULL THEN 'Unknown author' ELSE name END,
                title FROM Books LEFT OUTER JOIN Authors
            ON Books.author = Authors.id;
      name      |                 title
----------------+----------------------------------------
 Ève Demazière  | Les cultures noires d'Amérique Centrale
 Alain Damasio  | La horde du contrevent
 Fred Vargas    | Pars vite et reviens tard
 Fred Vargas    | L'homme à l'envers
 Unknown author | Bible
(5 rows)
```

Et si on veut garder, non pas les livres sans auteur mais les personnes sans livres, on utilise une jointure externe à droite ou bien, tout simplement, on inverse les tables :

```
essais=> SELECT name, title FROM Authors LEFT OUTER JOIN Books
                ON Books.author = Authors.id;
        name         |                 title
---------------------+----------------------------------------
 Stéphane Bortzmeyer |
 Fred Vargas         | Pars vite et reviens tard
 Fred Vargas         | L'homme à l'envers
 Ève Demazière       | Les cultures noires d'Amérique Centrale
 Alain Damasio       | La horde du contrevent
(5 rows)
```

Donc, contrairement à ce que présente l'article d'Atwood déjà cité, `INNER JOIN` n'est pas une intersection et `OUTER JOIN` n'est pas une union (SQL a des opérateurs pour ces opérations ensemblistes mais je ne les utilise pas ici). Les jointures créent des nouveaux tuples, elles ne sélectionnent pas des tuples existants.

Et si on veut aussi bien les livres sans auteurs que les gens qui n'ont pas écrit de livre ? C'est le rôle de la jointure externe complète, `FULL OUTER JOIN` :

```
essais=> SELECT name, title FROM Books FULL OUTER JOIN Authors ON Books.author = Authors.id;
        name         |                  title
---------------------+-----------------------------------------
 Stéphane Bortzmeyer |
 Fred Vargas         | Pars vite et reviens tard
 Fred Vargas         | L'homme à l'envers
 Ève Demazière       | Les cultures noires d'Amérique Centrale
 Alain Damasio       | La horde du contrevent
                     | Bible
(6 rows)
```

Et enfin, pour un exemple réel, emprunté à DNSmezzo <http://www.dnsmezzo.net/>, une jointure externe un peu plus compliquée. La table DNS_packets contient des paquets DNS, la table DNS_types, la correspondance entre le numéro de type d'enregistrement, contenu dans la requête, et des chaînes de caractères mnémoniques comme NAPTR ou SRV. On fait une sous-requête SQL pour ne garder que les paquets utiles, puis une jointure externe (car certains paquets contiennent des requêtes pour des types qui ne sont pas enregistrés dans la base IANA <https://www.iana.org/assignments/dns-parameters> et sont donc absents de la table DNS_types) :

```
dnsmezzo=> SELECT (CASE WHEN type IS NULL THEN qtype::TEXT ELSE type END),
                  meaning,
                  count(Results.id) AS requests
               FROM (SELECT id, qtype FROM dns_packets WHERE
                                (file=5 or file=13) AND query) AS
Results
                 LEFT OUTER JOIN DNS_types ON qtype = value
              GROUP BY qtype, type, meaning ORDER BY requests desc;
  type  |                meaning                | requests
--------+---------------------------------------+----------
 MX     | mail exchange                         |   983180
 A      | a host address                        |   847228
 AAAA   | IP6 Address                           |   129656
 NS     | an authoritative name server          |    13583
 SOA    | marks the start of a zone of authority |    10562
 TXT    | text strings                          |    10348
 255    |                                       |     9125
 38     |                                       |     8440
 SRV    | Server Selection                      |     3300
 SPF    |                                       |      677
 PTR    | a domain name pointer                 |      384
 CNAME  | the canonical name for an alias       |      351
 DNSKEY | DNSKEY                                |      323
 0      |                                       |       39
 26226  |                                       |       11
 NAPTR  | Naming Authority Pointer              |       11
 HINFO  | host information                      |        7
 NSEC   | NSEC                                  |        7
 8808   |                                       |        1
 14184  |                                       |        1
 3840   |                                       |        1
 54312  |                                       |        1
 13203  |                                       |        1
(23 rows)
```