КАК СТАТЬ АВТОРОМ 💰 Зарплаты IT-специалистов Поехали в гик-трип по Кал…

varanio
6 июн 2019 в 11:35

Понимание джойнов сломано. Продолжение. Попытка альтернативной визуализации

2 мин **61K**

Разработка веб-сайтов*, PostgreSQL*, Программирование*, SQL*

> Многие из вас читали предыдущую статью про то, как неправильная визуализация для объяснения работы JOIN-ов в некоторых случаях может запутать. Круги Венна не могут полноценно проиллюстрировать некоторые моменты, например, если значения в таблице повторяются.

При подготовке к записи шестого выпуска подкаста "Цинковый прод" (где мы договорились обсудить статью) кажется удалось нащупать один интересный вариант визуализации. Кроме того, в комментариях к изначальной статье тоже предлагали похожий вариант.

Все желающие приглашаются под кат

Итак, визуализация. Как мы выяснили в комментах к предыдущей статье, join — это скорее декартово произведение, чем пересечение. Если посмотреть, как иллюстрируют декартово произведение, то можно заметить, что зачастую это прямоугольная таблица, где по одной оси идет первое отношение, а по другой — второе. Таким образом элементы таблицы будут представлять собой все комбинации всего.

Сложно абстрактно это нарисовать, поэтому придется на примере.

Допустим, у нас есть две таблицы. В одной из них

```
id
--
1
1
6
5
```

В другой:

```
id
--
1
1
2
3
5
```

Сразу disclaimer: я назвал поле словом "id" просто для краткости. Многие в прошлой статье возмущались, как это так — id повторяются, безобразие. Не стоит сильно переживать, ну представьте, например, что это таблица с ежедневной статистикой, где для каждого дня и каждого юзера есть данные по посещению какого-нибудь сайта. В общем, не суть.

Итак, мы хотим узнать, что же получится при различных джойнах таблиц. Начнем с CROSS JOIN:

CROSS JOIN

```
SELECT t1.id, t2.id
FROM t1
    CROSS JOIN t2
```

CROSS JOIN — это все все возможные комбинации, которые можно получить из двух таблиц.

Визуализировать это можно так: по оси x — одна таблица, по оси y — другая, все клеточки внутри (выделены оранжевым) — это результат

	1	1	2	3	5
1	1, 1	1, 1	1, 2	1, 3	1, 5
1	1, 1	1, 1	1, 2	1, 3	1, 5
6	6, 1	6, 1	6, 2	6, 3	6, 5
5	5, 1	5, 1	5, 2	5, 3	5, 5

INNER JOIN

INNER JOIN (или просто JOIN) — это тот же самый CROSS JOIN, у которого оставлены только те элементы, которые удовлетворяют условию, записанному в конструкции "ON". Обратите внимание на ситуацию, когда записи дублируются — результатов с единичками будет четыре штуки.

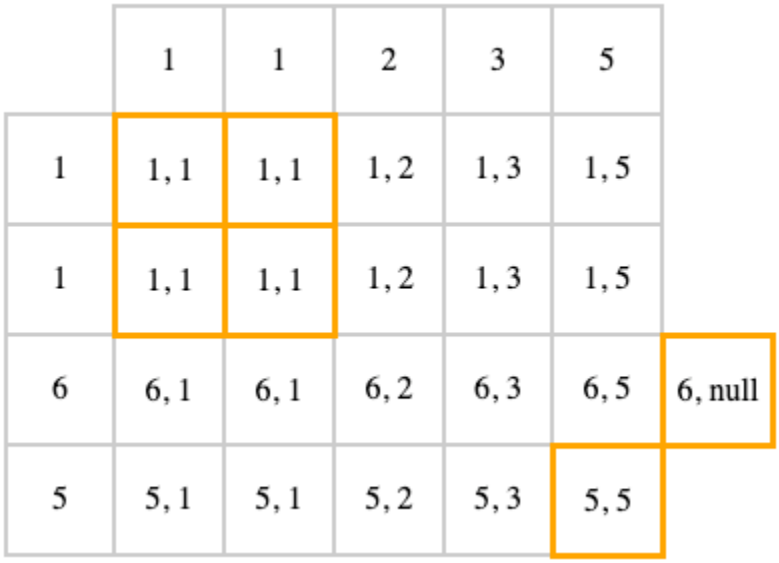
```
SELECT t1.id, t2.id
FROM t1
    INNER JOIN t2
        ON t1.id = t2.id
```

	1	1	2	3	5
1	1, 1	1, 1	1, 2	1, 3	1, 5
1	1, 1	1, 1	1, 2	1, 3	1, 5
6	6, 1	6, 1	6, 2	6, 3	6, 5
5	5, 1	5, 1	5, 2	5, 3	5, 5

LEFT JOIN

LEFT OUTER JOIN (или просто LEFT JOIN) — это тоже самое, что и INNER JOIN, но дополнительно мы добавляем null для строк из первой таблицы, для которой ничего не нашлось во второй

```
SELECT t1.id, t2.id
FROM t1
    LEFT JOIN t2
        ON t1.id = t2.id
```



RIGHT JOIN

RIGHT OUTER JOIN (или RIGHT JOIN) — это тоже самое, что и LEFT JOIN, только наоборот. Т.е. это INNER JOIN + null для строк из второй таблицы, для которой ничего не нашлось в первой

```
SELECT t1.id, t2.id
FROM t1
    RIGHT JOIN t2
        ON t1.id = t2.id
```

	1	1	2	3	5
1	1, 1	1, 1	1, 2	1, 3	1, 5
1	1, 1	1, 1	1, 2	1, 3	1, 5
6	6, 1	6, 1	6, 2	6, 3	6, 5
5	5, 1	5, 1	5, 2	5, 3	5, 5
			null, 2	null, 3	

→ Поиграть с запросами можно здесь

Выводы

Вроде бы получилась простая визуализация. Хотя в ней есть ограничения: здесь показан случай, когда в ON записано равенство, а не что-то хитрое (любое булево выражение). Кроме того не рассмотрен случай, когда среди значений таблицы есть null. Т.е. это всё равно некоторое упрощение, но вроде бы получилось лучше и точнее, чем круги Венна.

Больше полезного можно найти на telegram-канале о разработке "Cross Join", где мы обсуждаем базы данных, языки программирования и всё на свете!

Теги: join, left join, inner join, right join, круги Венна

Хабы: Разработка веб-сайтов, PostgreSQL, Программирование, SQL

+29 270 29