**SQL join**

Присоединение таблиц в запросах — это базовый инструмент в работе с базами данных. Давайте рассмотрим какие присоединения (JOIN) бывают, и что от этого меняется в результатах запроса.

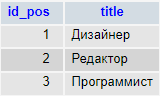
Для начала создадим две таблицы, над которыми будем проводить опыты. Это таблица с именами сотрудников и словарь с перечнем должностей.

**Persons** (Сотрудники)



Столбец *position\_ref* (от «position reference») это ссылка на следующую таблицу, где перечислены должности сотрудников.

**Positions** (должности)



Т.е. чтобы узнать должность сотрудника в таблице *Persons*, нужно присоединить соответствующие данные из таблицы  *Positions*, связывая их по значениям *position\_ref* и *id\_pos*.

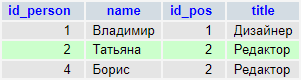
Далее мы рассмотрим все варианты присоединений. Данные специально подобраны так, чтобы продемонстрировать отличия в результатах разных запросов.

**INNER JOIN**

Внутреннее присоединение. Равносильно просто JOIN или CROSS JOIN (верно для MYSQL, в стандарте SQL INNER JOIN не эквивалентен синтаксически CROSS JOIN, т.к. используется с выражением ON).

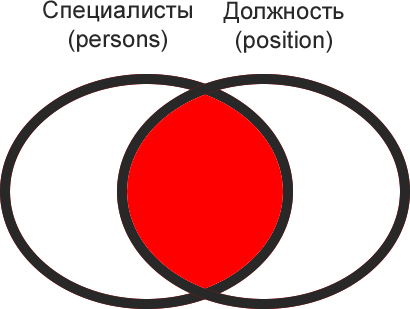
|  |
| --- |
| *SELECT id\_person, name, id\_pos, title*  *FROM `persons`*  *INNER JOIN `positions` ON id\_pos = position\_ref* |

Такое присоединение покажет нам данные из таблиц, только если условие связывания соблюдается — т.е. для сотрудника указан существующий в словаре идентификатор должности.



Если поменять порядок соединения таблиц — получим тот же результат.

Условно представим себе эти таблицы, как пересекающиеся множества, где пересечение — это наличие связи между таблицами. Получим картинку:



Далее проследим как получить разные части (подмножества) данного множества.

**OUTER JOIN**

Внешнее присоединение. Различают LEFT OUTER JOIN и RIGHT OUTER JOIN, и обычно опускают слово «OUTER».

Внешнее присоединение включает в себя результаты запроса INNER и добавляются «неиспользованные» строки из одной из таблиц. Какую таблицу использовать в качестве «добавки» — указывает токен LEFT или RIGHT.

**LEFT JOIN**

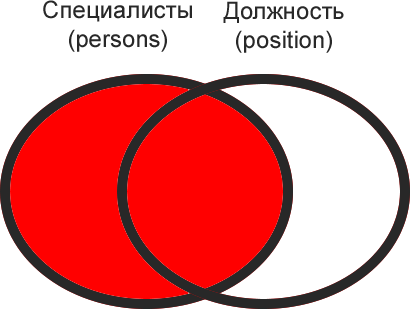
Внешнее присоединение «слева».

|  |
| --- |
| *SELECT id\_person, name, id\_pos, title*  *FROM `persons`*  *LEFT OUTER JOIN `positions` ON id\_pos = position\_ref* |



«Левая» таблица persons, содержит строку id\_person#3 — «Александр», где указан идентификатор должности, отсутствующей в словаре. Мы увидим все записи из «левой» таблицы, тогда как правая будет присоединена по возможности.

На картинке это можно показать вот так:



**RIGHT JOIN**

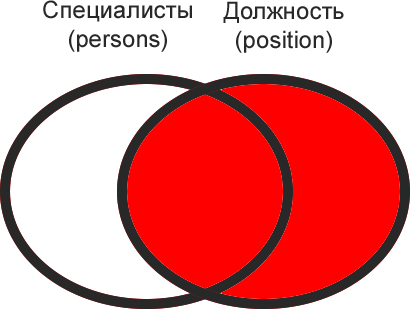
Присоединение «справа».

|  |
| --- |
| *SELECT id\_person, name, id\_pos, title*  *FROM persons*  *RIGHT OUTER JOIN positions ON id\_pos = position\_ref* |



Тут данные из «левой» таблицы присоединяются к «правой».

Словарь должностей (правая таблица) содержит неиспользуемую запись с id\_pos#3 — «программист». Теперь она попала в результат запроса.



**Полное множество**

MySQL не знает соединения FULL OUTER JOIN. Что если нужно получить полное множество?



**Первый способ** — объединение запросов LEFT и RIGHT.

|  |
| --- |
| *(SELECT id\_person, name, id\_pos, title*  *FROM persons*  *LEFT OUTER JOIN positions ON id\_pos = position\_ref)*    *UNION*    *(SELECT id\_person, name, id\_pos, title*  *FROM persons*  *RIGHT OUTER JOIN positions ON id\_pos = position\_ref)* |



При таком вызове UNION, после слияния результатов, SQL отсечет дубли (как DISTINCT). Для отсечения дублей SQL прибегает к сортировке. Это может сказываться на быстродействии.

**Второй способ** — объединение LEFT и RIGHT, но в одном из запросов мы исключаем часть, соответствующую INNER. А объединение задаём как UNION ALL, что позволяет движку SQL обойтись без сортировки.

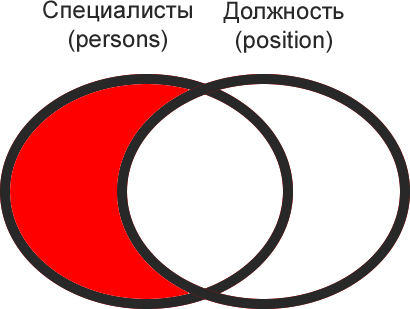
|  |
| --- |
| *(SELECT id\_person, name, id\_pos, title*  *FROM persons*  *LEFT OUTER JOIN positions ON id\_pos = position\_ref)*    *UNION ALL*    *(SELECT id\_person, name, id\_pos, title FROM persons*  *RIGHT OUTER JOIN positions ON id\_pos = position\_ref*  *WHERE id\_person is NULL)* |

Этот пример показывает нам как исключить пересечение и получить только левую или правую часть множества.

**Левое подмножество**

LEFT JOIN ограничиваем проверкой, что данных из второй таблицы нет.

|  |
| --- |
| *SELECT id\_person, name, id\_pos, title*  *FROM persons*  *LEFT OUTER JOIN positions ON id\_pos = position\_ref*  *WHERE id\_pos is NULL* |



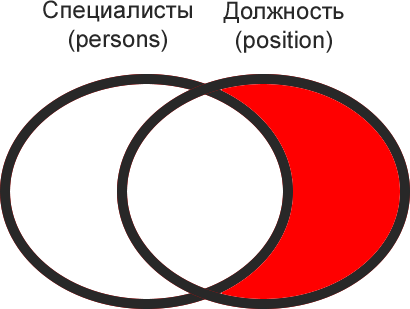
В нашем примере — это специалисты, у которых не задана должность или нет должности с указанным ключом.

https://shra.ru/wp-content/uploads/2021/09/left-join-strict-query.png

**Правое подмножество**

Аналогично выделяем правую часть.

|  |
| --- |
| *SELECT id\_person, name, id\_pos, title*  *FROM persons*  *RIGHT OUTER JOIN positions ON id\_pos = position\_ref*  *WHERE id\_person is NULL* |



https://shra.ru/wp-content/uploads/2021/09/right-join-strict-query.png

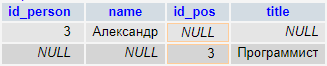
В нашем случае получим должности, которые никому не назначены.

**Всё кроме пересечения**

Остался один вариант, тот, когда исключено пересечение множеств. Его можно сложить из двух предыдущих запросов через UNION ALL (т.к. подмножества не пересекаются).

|  |
| --- |
| *(SELECT id\_person, name, id\_pos, title*  *FROM persons*  *LEFT OUTER JOIN positions ON id\_pos = position\_ref*  *WHERE id\_pos is NULL)*    *UNION ALL*    *(SELECT id\_person, name, id\_pos, title*  *FROM persons*  *RIGHT OUTER JOIN positions ON id\_pos = position\_ref*  *WHERE id\_person is NULL)* |

Это запрос соберет все случаи, когда по какой-то причине данные из таблиц не связаны.



А графически такое объединение выглядит следующим образом:

