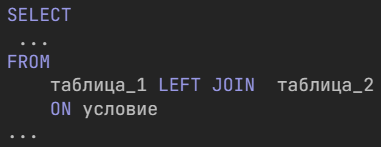
LEFT JOIN соединяет две таблицы.

Порядок таблиц для оператора **важен**, поскольку оператор не является симметричным.



Результат запроса формируется так:

- в результат включается внутреннее соединение (INNER JOIN) первой и второй таблицы в соответствии с условием;

- затем в результат добавляются те записи первой таблицы, которые не вошли во внутреннее соединение на шаге 1, для таких записей соответствующие поля второй таблицы заполняются значениями NULL.

Соединение RIGHT JOIN действует аналогично, только в пункте 2 первая таблица меняется на вторую и наоборот. ← **Если ни хера не понятно читай ниже**.

Пример: Вывести название всех книг каждого автора, если книг некоторых авторов в данный момент нет на складе – вместо названия книги указать Null.

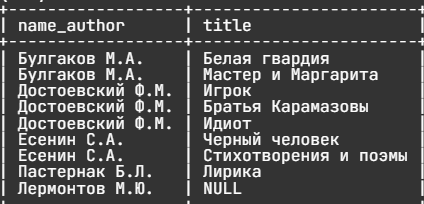
**SELECT name\_author, title**

**FROM author LEFT JOIN book**

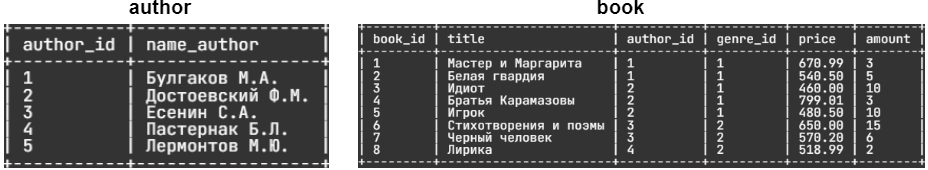
**ON author.author\_id = book.author\_id**

**ORDER BY name\_author;**

**Результат.**



Разберёмся: Есть 2 таблицы



Берётся первая строка из левой таблицы и первая строка из правой таблицы. Смотрим **author.author\_id = book.author\_id** если да, тогда выводим данную строку (в смысле то, что указано в SELECT-е), т.е. всё тоже самое что и в inner join.



далее сравниваем всё туже первую строку из левой таблицы и **вторую** строку правой таблицы(так как первую строку правой таблицы мы обработали) и так до конца правой таблицы, обратно-же всё как и в INNER JOIN.

**Ключевая особенность этого JOIN-a, - когда в левой таблице есть значение author.author\_id а в правой колонке такого значения в принципе не существует, тогда такой ряд тоже выводится но в результате правой таблицы вносится NULL.**

В нашем примере в левой таблице есть **author.author\_id = 5** но в правой таблице (book) в **book.author\_id** такого значения нет и так как оператор у нас LEFT JOIN то мы выводим строку с левой таблицы а с правой таблицы, так как там нет такого значения, мы подставляем NULL. (смотри результат выше)

Т.е. при использовании этого оператора (**JOIN**) основное значение имеет расположение таблиц в запросе после FROM (т.е. как они (таблицы) располагаются по отношению к этому оператору (JOIN-у), та таблица которая указывается сразу после FROM это (левая таблица с точки зрения оператора JOIN, так как она слева от него) или первая, (если с точки зрения поочерёдности), а таблица указанная после оператора JOIN это правая таблица, или вторая. А то как будет написано условие ON вообще не имеет значения с точки зрения соединения таблиц.

Используя LEFT JOIN мы основной таблицей считаем левую таблицу (в примере это author) и все ряды из неё которые соответствуют условию будут выведены, а из правой таблицы если какого то значения не будет то оно будет заменено на NULL.

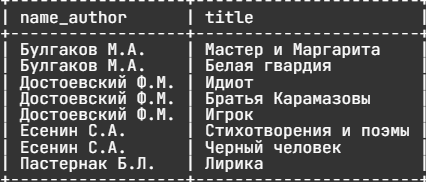
Если взять те же таблицы и поменять их местами в разделе FROM

**SELECT name\_author,title**

**FROM book LEFT JOIN author**

**ON book.author\_id = author.author\_id**

Результат:



Получается что теперь основная таблица поменялась с author на book, так как теперь она расположена слева от JOIN-а. И теперь все строки соответствующие условию из book будут выведены, и по ним будет проверятся выборка и если во второй (правой) таблице будут ещё записи которых нет в левой таблице то они выводится не будут.

Т.е. тот же алгоритм: берётся первая строка с левой таблицы и первая строка из правой таблицы

если **book.author\_id = author.author\_id** то выводим то что просят в SELECT-е. Далее оставляем первую строку из левой таблицы, а из правой таблицы берём вторую строку и если **book.author\_id = author.author\_id то** выводим, если нет проверяем следующую строку из правой таблицы пока она не закончится. Затем берётся вторая строка из левой таблицы и т.д..

Пример: Вывести все жанры, которые не представлены в книгах на складе.

**SELECT name\_genre**

**FROM genre LEFT JOIN book**

**ON genre.genre\_id = book.genre\_id**

**WHERE book.genre\_id IS NULL**

**Запросы для нескольких таблиц (join) с вложенными запросами**

В запросах, построенных на нескольких таблицах, можно использовать вложенные запросы. Вложенный запрос может быть включен: после ключевого слова SELECT, после FROM и в условие отбора после WHERE (HAVING).

Пример: Вывести авторов, общее количество книг которых на складе максимально.

**SELECT name\_author, SUM(amount) as Количество**

**FROM**

**author INNER JOIN book**

**on author.author\_id = book.author\_id**

**GROUP BY name\_author**

**HAVING SUM(amount) =**

**(/\* вычисляем максимальное из общего количества книг каждого автора \*/**

**SELECT MAX(sum\_amount) AS max\_sum\_amount**

**FROM**

**(/\* считаем количество книг каждого автора \*/**

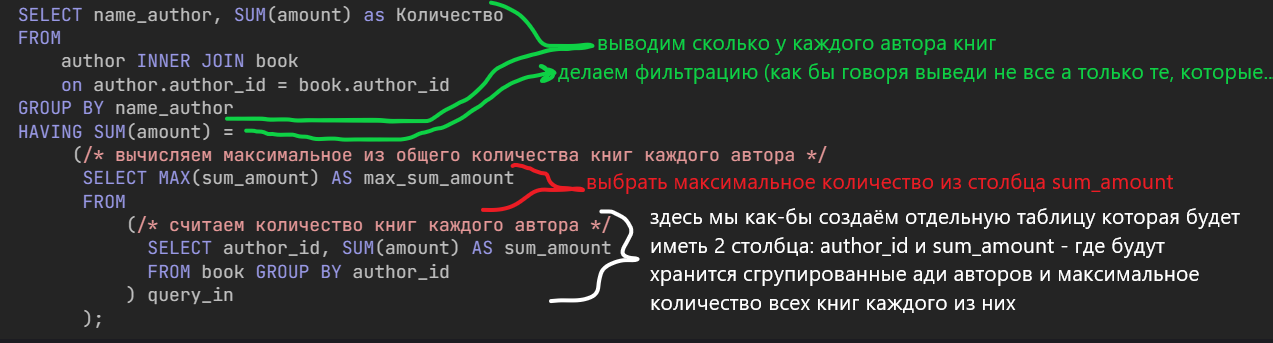
**SELECT author\_id, SUM(amount) AS sum\_amount**

**FROM book GROUP BY author\_id**

**) query\_in**

**);**

ПОЯСНЕНИЯ:



Наверно следует рассматривать данный запрос с самого вложенного (с белыми скобками на ресунке), который возвращает таблицу(грубо говоря). Эта таблица имеет только 2 столбца со значениями. В среднем по вложенности запросе (красный) мы просто находим максимальное значение в одном из столбцов этой таблицы, это значение =23. Ано то нам и надо для условия в самом внешнем запросе. Теперь по сути запрос выглядит вот так:

SELECT name\_author, SUM(amount) as Количество

FROM

author INNER JOIN book

on author.author\_id = book.author\_id

GROUP BY name\_author

HAVING SUM(amount) = 23

