новая тема будет про соединения таблиц. Про то, как выгружать данные из разных таблиц. есть несколько типов соединений:

1. внутреннее соединение. INNER JOIN

2.LEFT и RIGHT JOIN -

3. FULL JOIN

4. CROSS JOIN - декартово соединение

5. SELF JOIN

1.

пока предварительно, подробнее будет потом. Внутреннее соединение доступно, когда есть несколько таблиц, например, книга и издатель, и пять книг издают два издателя. Тогда происходит соединения, где книги - это главное, а вот издатели, т.к. их меньше - уже дополняют, как-то так, хз.

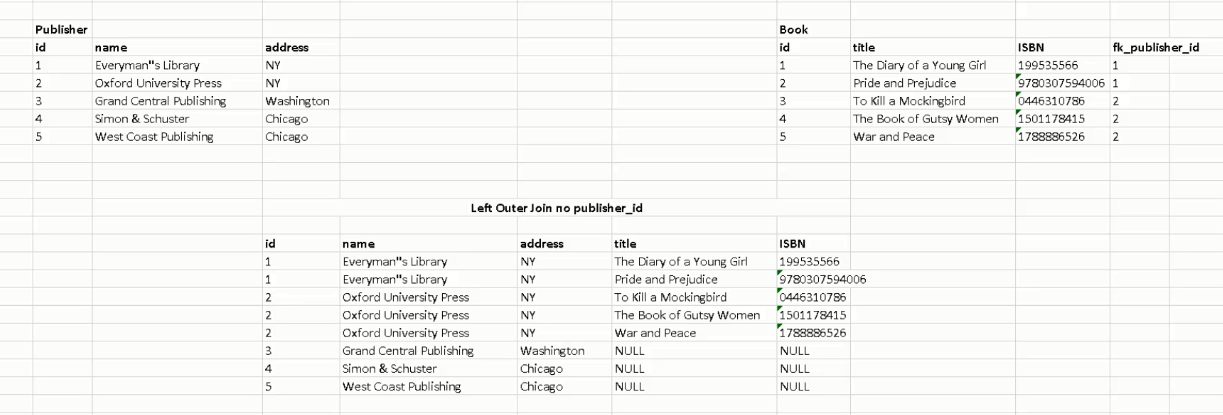
внизу на таблице видно, что и как соотносится по внешнему и внутреннему ключам



2.

про правое и левое соединение, обычно табличка слева - это №1, а справа - №2. Их можно соединять, менять местами, тогда у них и названия будут меняться. **То, где находятся таблица, непосредственно влияет на результат.**

на примере ниже видно, что левое соединение прошло, всё, что надо, было соединено, однако для некоторых издателей не было найдено совпадений по внешнему ключу во второй таблице, поэтому их столбцы пустые

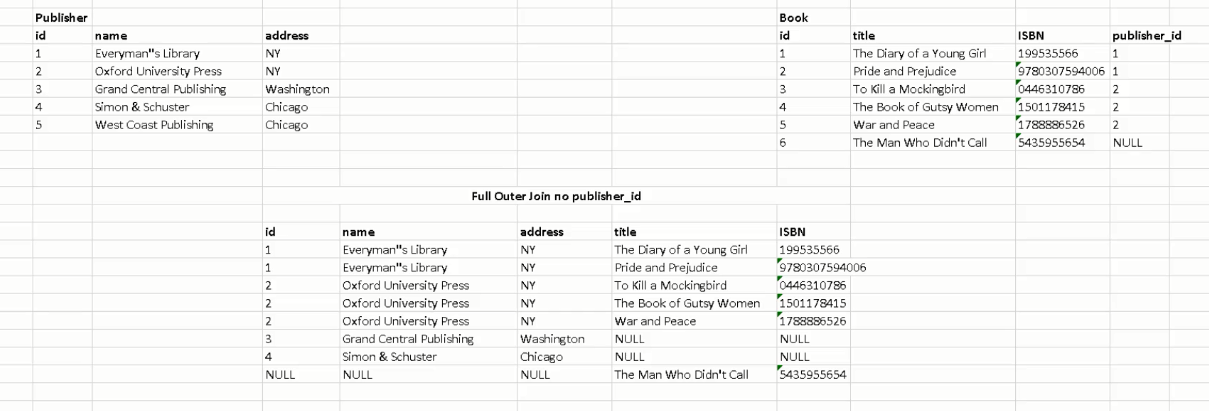


3. что касается правостороннего соединения, когда правая табличка основная и используются все ее строки

здесь видно, что они отработала также как и первое соединение, под все книги были найдены издатели, а остальные не были учтены, т.к. правая таблица ограничена всего двумя ключами 1 и 2.  
но как мб будет правильнее делать: нужно ставить **левой** ту таблицу, которую мы хотим использовать основной, чьи все данные нам будут нужны. запрос делается по книгам, и т.к. там только 2 ключа 1 и 2, он берет только два издателя



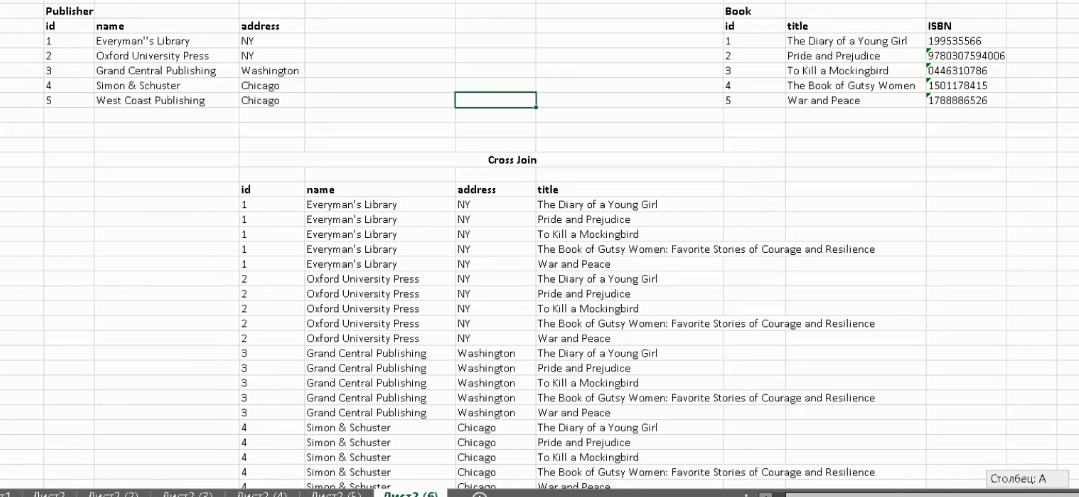
4. FULL JOIN - просто объединяет две таблицы. одновременно и левое, и правое соединение. Создает одну большую таблицу, в которой есть поля и из первой, и из второй таблиц. Где куда чего не хватает - просто ставит NULL.



5. CROSS JOIN - декартово соединение. Само по себе декартово соединение это пересечение двух множеств в различных вариациях. Например, х = 1, 2 и у = 3, 4

пересечениями будут : 1-3, 1-4, 2-3, 2-4 по **х**, и 3-1, 3-2, 4-1, 4-2 по **у**

**а как тут - видно, что каждому издателю он пихает книгу.** пригождаются обычно при рассмотрении различных стат данных



**INNER JOIN**

Здесь надо знать порядок запросов: сначала мы делаем селект, потом откуда, затем уже идут соединения, первый фильтр, группировка, второй фильтр, сортировка

SELECT products.product\_name, suppliers.company\_name, units\_in\_stock

FROM suppliers

INNER JOIN products ON products.supplier\_id = suppliers.supplier\_id

ORDER BY units\_in\_stock ASC

здесь мы выбираем имя продуктов, компаний и количество товаров,

надо выбрать таблицу,

а для второй таблицы мы уже джойн делаем, указываем, что у двух разных таблиц одинаковые столб\_ID

сортируем по какой-нибудь херне и прочее

**другая задача**: посчитать количество товаров в продаже по категориям , для этого надо соединить категории с продуктами

SELECT categories.category\_name, SUM(units\_in\_stock)

FROM products

INNER JOIN categories ON products.category\_id = categories.category\_id

GROUP BY categories.category\_name

ORDER BY SUM(units\_in\_stock) ASC

**Задача 3:** тут надо сделать вывод того, на какую сумму продаются товары в отдельных категориях, при этом только те, которые именно в наличии есть сейчас ите, чья сумма больше 5000 уе

SELECT categories.category\_name, SUM(units\_in\_stock \* unit\_price)

FROM products

INNER JOIN categories ON products.category\_id = categories.category\_id

WHERE discontinued <> 1

GROUP BY categories.category\_name

HAVING SUM(units\_in\_stock \* unit\_price) > 5000

ORDER BY SUM(units\_in\_stock \* unit\_price) ASC

**Задача 4:** на каких работниках завязаны заказы

SELECT order\_id, customer\_id, first\_name, last\_name, title

FROM orders

INNER JOIN employees ON employees.employee\_id = orders.employee\_id

LIMIT 10

**Задача 5:** здесь уже нужно выбрать данные из трех таблиц, в селекте также перечисляем, какие поля мы хотим видеть, просто делаем два соединения

SELECT order\_date, ship\_country, products.unit\_price, quantity, discount

FROM orders

INNER JOIN order\_details ON order\_details.order\_id = orders.order\_id

INNER JOIN products ON order\_details.product\_id = products.product\_id

WHERE discount <> 0

ORDER BY ship\_country, discount DESC

**Задача 6:** просто вывести дохрена всякого

SELECT contact\_name, company\_name, phone, first\_name, last\_name, title,

order\_date, product\_name, ship\_country, products.unit\_price, quantity, discount

FROM orders

JOIN order\_details ON order\_details.order\_id = orders.order\_id

JOIN products ON order\_details.product\_id = products.product\_id

JOIN customers ON customers.customer\_id = orders.customer\_id

JOIN employees ON employees.employee\_id = orders.employee\_id

WHERE discount <> 0

LIMIT 10

**LEFT RIGHT JOIN**

**Если всем ключам есть соответствие, то LEFT будет равен INNER.**

Для чего нам может потребоваться ЛЕФТ джойн? Применительно к нашим примерам, вдруг нам необходимо узнать, у каких компаний-покупателей на данный момент нет заказов

SELECT company\_name, order\_id

FROM customers

LEFT JOIN orders on customers.customer\_id = orders.customer\_id

WHERE order\_id IS NULL

Если мы делаем INNER, то те конторы, которые не имеют order\_id, будут отсекаться, но если мы делаем LEFT, то результат будет такой

"Paris spécialités" null

"FISSA Fabrica Inter. Salchichas S.A." null

Имя компании относится к покупателям, и именно из него берется инфа, поэтому он находится слева, потому что в запросе он находится выше, а чтение идет слева направо. Значит, вторая таблица с заказами и их ID уже находится справа. И если на этом вызове сделать RIGHT JOIN, то результатов не будет. Т.к. уже конторы без ID уже будут отсекаться

Однако, если мы поменяем местами таблицы в запросе, то результаты опять появятся

**Есть ли работники, которые не обрабатывают никакие заказы?**

SELECT COUNT(\*)

FROM employees

LEFT JOIN orders on employees.employee\_id = orders.employee\_id

WHERE order\_id IS NULL

Ничего не найдет, потому что все работники заняты

**CROSS JOIN**

**–** там отсутствует часть ON,

SELECT COUNT(\*)

FROM employees

CROSS JOIN orders

**FULL JOIN**

**– хз, разберемся ещё.**

**SELF JOIN**

Обычно требуется для того, чтобы построить некоторую иерархию, когда есть некоторое иерархическое отношение между данными. Но тут надо, чтобы таблица в себе могла ссылаться на саму себя

Создадим таблицу.

CREATE TABLE employee

(

employee\_id INT primary key,

first\_name varchar(255) not null,

last\_name varchar(255) not null,

manager\_id int,

FOREIGN KEY (manager\_id) REFERENCES employee(employee\_id)

)

Тут что к чему: у каждого работника есть свой айди, имя, фамилия и айди его т.с. начальника, но т.к. начальник сам является работником, и у него есть свой айди работника, у каждого работника есть внешний ключ, который ссылается по manager\_id на менеджера как на работника.

Вот у нас есть работники

insert into employee

values

(1,'Al','Kir',NULL),

(2,'So','Fa',1),

(3,'Do','Re',1),

(4,'El','Ray',2),

(5,'Tho','Mas',3),

(6,'I','Van',3),

(7,'Van','Gog',3),

(8,'Van','Damm',3)

Как нам вывести работника и рядом менеджера этого работника?

SELECT e.first\_name || ' ' || e.last\_name as employee,

m.first\_name || ' ' || m.last\_name as manager

FROM employee e

LEFT JOIN employee m ON m.employee\_id = e.manager\_id

ORDER BY manager DESC

Здесь мы придумываем псевдоним для работника, затем придумываем псевдоним для начальника. Т.к. нам надо знать у какого работника какой начальник, то именно работник и будет находиться слева, а лефтджойнит будем начальника. Но что тут важно. Важно то, что рук ссылается на айди работника, а работник ссылается на айди начальника. И мы видим вот такой результат

"Al Kir" NULL

"El Ray" "So Fa"

"Van Damm" "Do Re"

"I Van" "Do Re"

"Van Gog" "Do Re"

"Tho Mas" "Do Re"

"Do Re" "Al Kir"

"So Fa" "Al Kir"

**USING AND NATURAL JOIN**

Это типа синтаксический сахар

USING используется для того, чтобы указать, по каким столбцам мы хотим соединить таблицы. Один из предыдущих примеров вместо **такой** записи:

SELECT COUNT(\*)

FROM employees

LEFT JOIN orders on employees.employee\_id = orders.employee\_id

WHERE order\_id IS NULL

Можно записать **так:**

SELECT COUNT(\*)

FROM employees

LEFT JOIN orders USING(employee\_id)

WHERE order\_id IS NULL

**Соединить работников и заказы, используя айди работника**

**Псевдонимы – AS**

Когда нам надо что-то где-то сделать, например, посчитать количество работников в компании, мы может просто сделать так:

select COUNT(\*)

from employees

система выведет количество сотрудников, но при это столбик не будет красиво проименован, его название будет просто count, если делать чуть более явно, то можно использовать псевдоним

select COUNT(\*) as employee\_number

from employees

тогда система выведет столбик уже с нужным именем, и там будет указано число сотрудников.

Псевдонимы нельзя использовать в разделе WHERE и HAVUNG в связи с тем, что сначала идет калькулирование условий, а уже потом накладывание псевдонима, но можно в GROUP BY и ORDER BY

SELECT category\_id, SUM(unit\_price \* units\_in\_stock) as total\_price

from products

where discontinued <> 1

group by category\_id

having SUM(unit\_price \* units\_in\_stock) > 5000 - тут мы пишем именно сумму

order by total\_price

ЗАДАЧИ

1.

**Найти заказчиков и обслуживающих их заказы сотрудников таких, что и заказчики и сотрудники из города London, а доставка идёт компанией Speedy Express. Вывести компанию заказчика и ФИО сотрудника**.

select cus.company\_name, concat(first\_name, ' ', last\_name), cus.city

from orders as o

inner join employees as emp using(employee\_id)

inner join customers as cus using(customer\_id)

inner join shippers as s ON o.ship\_via = s.shipper\_id

where emp.city = 'London' and cus.city = 'London' and s.company\_name = 'Speedy Express'

тут cus.company\_name именно потому, что нам надо знать название компании, с городом тоже есть псевдоним, чтобы мы могли его просто увидеть.

Импортируем именно из заказов, -- мб потому, что у него есть и customer\_id и employee\_id, как более центральная таблица.

Джойним по ключам, придумывая псевдоним, а вот с шипером было не все так очевидно, потому что всего 6 компаний-перевозчиков, а в заказах столбец «отправка через» - всего три, поэтому неочевидно было, да и из-за разного наименования столбцов их просто не заджойнить через юзинг

2.

**--Найти активные (см. поле discontinued) продукты из категории Beverages и Seafood, которых в продаже менее 20 единиц.**

**--Вывести наименование продуктов, кол-во единиц в продаже, имя контакта поставщика и его телефонный номер.**

select product\_name, units\_in\_stock, contact\_name, phone, category\_name

from products

join categories using(category\_id)

join suppliers using(supplier\_id)

where discontinued <> 1 and units\_in\_stock < 20 and (category\_name in('Beverages', 'Seafood'))

ORDER BY units\_in\_stock

3 и 4

**-- Найти заказчиков, не сделавших ни одного заказа. Вывести имя заказчика и order\_id, а 4-я наоборот**

select company\_name, order\_id

from customers

LEFT JOIN orders on customers.customer\_id = orders.customer\_id

WHERE order\_id IS NULL

select company\_name, order\_id

from orders

RIGHT JOIN customers on customers.customer\_id = orders.customer\_id

WHERE order\_id IS NULL