***МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ***

***Федеральное государственное автономное***

***образовательное учреждение***

***высшего образования***

**«Национальный исследовательский ядерный университет «МИФИ»»**

**Кафедра №42 «Криптология и кибербезопасность»**

*ЛАБОРАТОРНАЯ РАБОТА №1-3:*

*«Сложные запросы на выборку. Соединения»*

Аверин Владислав

Группа Б19-505

Апрель, 2022

*Содержание*

[*1. Подзапросы* 3](#_Toc1)

[*2. Соединения* 4](#_Toc2)

[*3. Иерархические запросы* 9](#_Toc3)

[*4. Аналитические функции*  11](#_Toc4)

[*Выводы* 14](#_Toc5)

## *1. Подзапросы*

**1.1** Для начала найдем сотрудников, которые не связаны ни с одним делом (балласт, в общем):

***SELECT*** *\** ***FROM*** *Employees*

***WHERE*** *employee\_id* ***NOT IN***

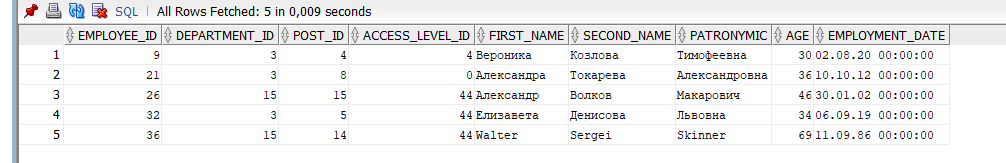
*(*

***SELECT DISTINCT*** *employee\_id* ***FROM*** *AssignedCases*

*)*

***ORDER BY*** *employee\_id;*

Вывод:



*\*(department 3 – это бухгалтерия, а 15 – высшие должностные лица)*

Усложним немного запрос, добавив, например, ограничение на стаж работы больше 5 лет (просто так):

***SELECT*** *\** ***FROM*** *Employees*

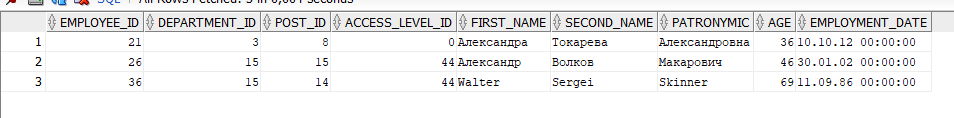
***WHERE*** *employee\_id* ***NOT IN***

*(*

***SELECT DISTINCT*** *employee\_id* ***FROM*** *AssignedCases*

*)* ***AND*** *( SYSDATE - employment\_date ) > 365\*5*

***ORDER BY*** *employee\_id;*



***1.2*** *«Найти людей, работающих в участке с самого "открытия" (т.е. те, у которых employment\_date меньше, чем у самого раннего дела в таблице Cases)»*

***SELECT*** *\** ***FROM*** *Employees*

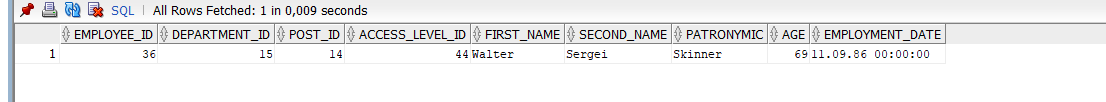
***WHERE*** *employment\_date <*

*(*

***SELECT******MIN****(start\_date)* ***FROM*** *Cases*

*);*

*Вывод:*



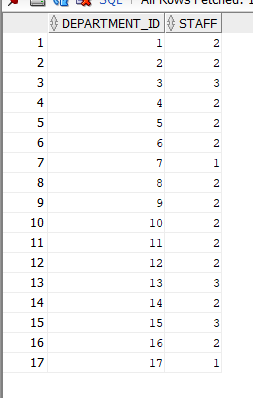
## *2. Соединения*

***2.1*** *Выведем статистику штата сотрудников по отделам (кол-во человек в каждом отделе):*

***SELECT*** *department\_id,* ***COUNT****(\*) AS staff* ***FROM*** *Employees*

***GROUP BY*** *department\_id*

***ORDER BY*** *department\_id;*

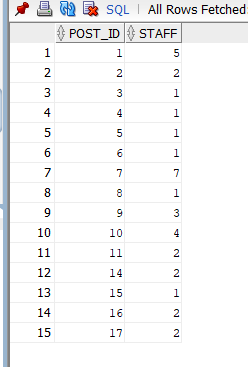


*Вывод:*

Note: такой запрос не будет показывать отсутствие штата у каких-то отделов (логично, ведь в таблице записей о них вообще нету). В данном случае этого не видно, т.к. во всех отделах имеются сотрудники. Однако попробуем написать ту же статистику по должностям:

***SELECT*** *post\_id,* ***COUNT****(\*) AS staff* ***FROM*** *Employees*

***GROUP BY*** *post\_id*



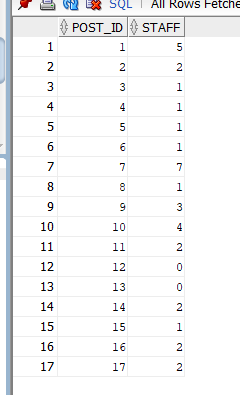
***ORDER BY*** *post\_id;*

Заметим, что в запросе отсутствуют записи post\_id = 12, 13. Исправим это, реализовав запрос через внешнее (left outer) соединение с таблицей Posts, чтобы в таблице были все должности (не забыв при этом, как обработать null-значения):

***SELECT*** *Posts.post\_id,* ***COUNT****(****CASE WHEN*** *employee\_id* ***IS NOT NULL THEN*** *0* ***END****)* ***AS*** *staff* ***FROM*** *Posts*

***LEFT JOIN*** *Employees* ***ON*** *employees.post\_id = posts.post\_id*

***GROUP BY*** *Posts.post\_id;*



*Получаем следующее:*

Как видим, напротив post\_id = 12 и 13 расположены верные значения.

(Не важно, какое число использовать росле 'then', т.к. count считает именно строки, а не значения атрибута. Если бы, к примеру, вместо COUNT было бы SUM, то надо было бы поставить 1)

Немного другой вариант запроса (который я хотел сделать первоначально), в котором сначала производится агрегирование, а уже затем - соединение с необходимой обработкой null значений:

***SELECT*** *Posts.post\_id,* ***COALESCE****(tabl1.res, 0)* ***AS*** *staff* ***FROM***

*(*

***SELECT*** *post\_id,* ***COUNT****(\*)* ***AS*** *res* ***FROM*** *Employees*

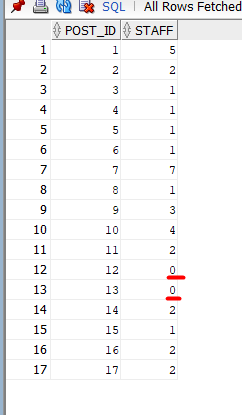
***GROUP BY*** *Employees.post\_id*

*) tabl1*

***RIGHT JOIN*** *Posts*

***ON*** *tabl1.post\_id = Posts.post\_id*

***ORDER BY*** *Posts.post\_id;*



Результат тот же, но второй вариант как по мне более громоздкий.

***2.2*** *Изменим условие п. 1.1 на то, что выбранные сотрудники вместо связи с каким-то делом (что фактически означает наличие сотрудника в таблице) не должны на данный момент вести никаких дел. То есть в данном случае нам необходима дополнительная информация из таблицы Cases (состояние дела), а не только найти все employee\_id из Assigned Cases:*

***SELECT*** *\** ***FROM*** *Employees*

***WHERE*** *employee\_id* ***NOT IN***

*(*

***SELECT DISTINCT*** *employee\_id*

***FROM*** *Cases*

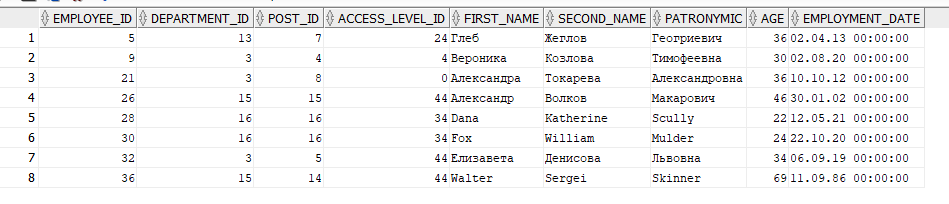
***INNER JOIN*** *AssignedCases* ***ON*** *AssignedCases.case\_id = Cases.case\_id*

***WHERE*** *status\_id = 1*

*)*

***ORDER BY*** *employee\_id;*

Вывод:



*Изменив немного запрос, найдем сотрудников, которые занимаются на данный момент не больше чем двумя делами одновременно. Для этого снова найдем сотрудников, которые нам не подходят, и вычтем их из всех сотрудников (используя MINUS вместо '****WHERE*** *employee\_id* ***NOT IN****' просто так, чтобы было):*

***SELECT*** *employee\_id* ***FROM*** *Employees*

***MINUS***

*(*

***SELECT*** *employee\_id* ***FROM***

*(*

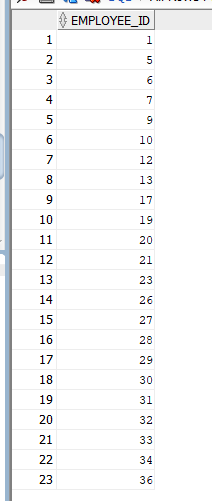
***SELECT*** *employee\_id,* ***COUNT****(\*)* ***AS*** *actual\_cases*

***FROM*** *Cases*

***INNER JOIN*** *AssignedCases* ***ON*** *AssignedCases.case\_id = Cases.case\_id*

***WHERE*** *status\_id = 1*

***GROUP BY*** *employee\_id*



*)*

***WHERE*** *actual\_cases > 2*

*)*

***ORDER BY*** *employee\_id;*

## *3. Иерархические запросы*

С ними были небольшие проблемы, т.к. в моей БД всунуть иерархическое наследование было некуда (в ту структуру, которая была у меня). Поэтому, я решил просто создать специально для этого типа запросов новую таблицу иерархической системы персонала (заодно дополнив БД новой инфой). Она включает в себя поля id (для уникальности строк, т.к. отношение таблицы к Posts будет 0..n), post\_id и pid (идентификатор родителя, на который ссылается данная должность в этой же таблице).

(Note: ниваажна, что оно может выглядеть не логичным; я попытался описать максимально правдоподобную структуру должностей; и я хз, нормально ли, что в одной таблице два внешних ключа, которые ссылаются на один и тот же атрибут).

*Создадим саму таблицу*

***CREATE TABLE*** *PeckingOrder*

*(*

*post\_id,*

*pid* ***NUMBER(2,0)****,*

***CONSTRAINT*** *PeckingOrder\_pk* ***PRIMARY KEY*** *(post\_id),*

***CONSTRAINT*** *PeckingOrder\_fk\_post*

***FOREIGN KEY*** *(post\_id)*

***REFERENCES*** *Posts(post\_id),*

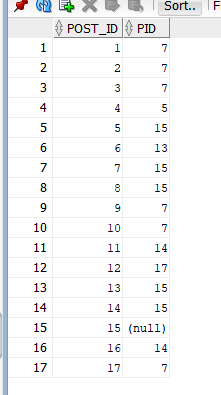
***CONSTRAINT*** *PeckingOrder\_fk\_pid*

***FOREIGN KEY*** *(pid)*

***REFERENCES*** *Posts(post\_id)*

*);*

*Заполним ее (я ее заполнял через sql developer):*



Собственно, сам запрос (идея "отрисовки" дерева через lpad честно и полностью скомунизжена с Хабра):

***SELECT LPAD****(' ', 3\*****LEVEL****)||post\_name* ***AS*** *pecking\_order*

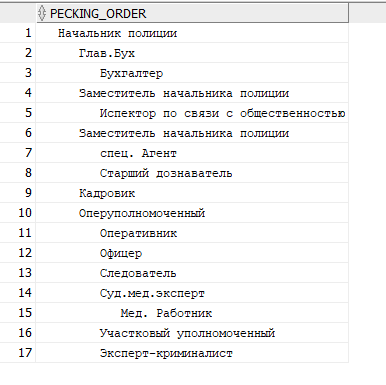
***FROM***

*(*

***SELECT*** *Posts.post\_id, PeckingOrder.pid, Posts.post\_name* ***FROM*** *Posts*

***INNER JOIN*** *PeckingOrder*

***ON*** *Posts.post\_id = PeckingOrder.post\_id*



*)*

***CONNECT BY PRIOR*** *post\_id = pid*

***START WITH*** *pid* ***IS NULL***

***ORDER SIBLINGS BY*** *post\_name;*

Все же проще было эту иерархию сделать напрямую в таблице Employees по id "босса" каждого сотрудника; однако, никто же не запрещает потом поменять таблицы (хотя, по сути оба варианта не особо полезны; кроме проверки того, какую должность надо давать сотруднику при повышении, я больше назначений ей не вижу). Поэтому, пусть пока останется она, и в будущем если что использую ее для того, чтобы можно было добавить pid в таблицу Employees и наглядно расставить их значения всем сотрудникам (в зависимости от должности).

## *4. Аналитические функции*

*Предположим, нам необходимо вывести вместе с информацией о сотруднике (будем во избежание загромождения выводить только employee\_id) дополнительную аналитику, например, кол-во сотрудников в его отделе. В данномм случае, можно написать подзапрос со статистикой каждого отдела, и соединить его с родительской таблицей Employees:*

***SELECT*** *employee\_id, Employees.department\_id, tabl2.staff* ***FROM*** *Employees*

***INNER JOIN***

*(*

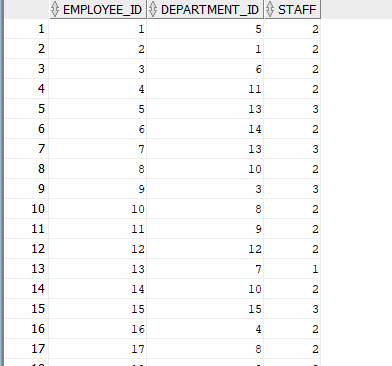
***SELECT*** *department\_id,* ***COUNT****(\*)* ***AS*** *staff* ***FROM*** *Employees*

***GROUP BY*** *department\_id*

*) tabl2*

***ON***

*Employees.department\_id = tabl2.department\_id;*



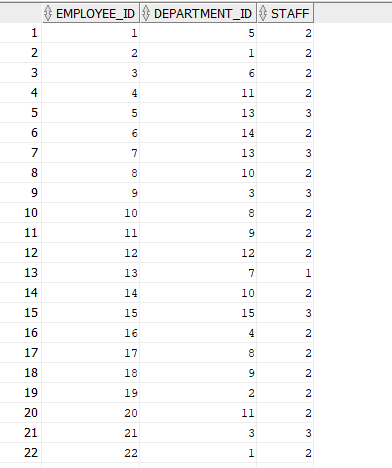
Однако выглядит это довольно... Громоздко. А предположим еще, что нам нужен не один аналитический столбец, а несколько с разными механизмами (и представляем нагромождение JOIN'ов). Чтобы облегчить себе жизнь, рационально использовать аналитические функции (ORDER BY делаем, чтобы порядок строк в запросе был такой же, как в предыдущем варианте):

***SELECT*** *Employee\_id, department\_id,*

***COUNT****(\*)* ***OVER*** *(****PARTITION BY*** *department\_id)* ***AS*** *staff*

***FROM*** *Employees*

***ORDER BY*** *employee\_id;*



Результаты эквивалентны, однако вторая запись более простая. Кроме того, возвращаясь к вопросу о нескольких аналитических столбцах, в одной выборке SELECT можно использовать несколько аналитических функций. Выведем, к примеру, с размером штата сотрудников отделов, кол-во сотрудников соответствующих должностей и средний возраст сотрудников отдела:

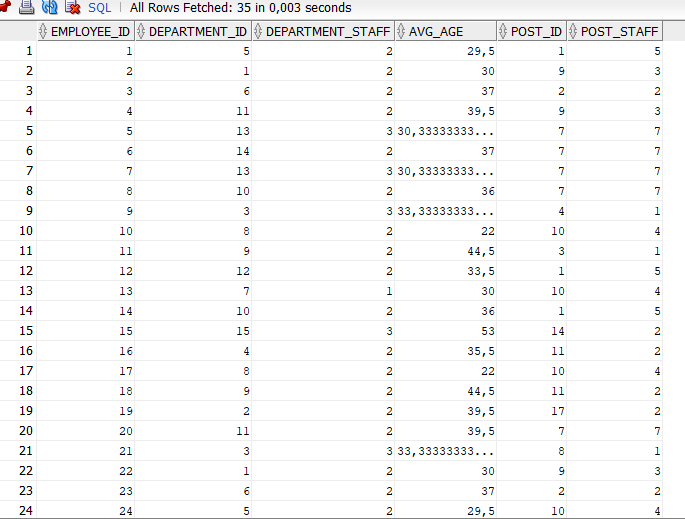
***SELECT*** *Employee\_id, department\_id,*

***COUNT****(\*)* ***OVER*** *(****PARTITION BY*** *department\_id)* ***AS*** *department\_staff,*

***AVG****(age)* ***OVER*** *(****PARTITION BY*** *department\_id)* ***AS*** *AVG\_age,*

*post\_id,* ***COUNT****(\*)* ***OVER*** *(****PARTITION BY*** *post\_id)* ***AS*** *post\_staff* ***FROM*** *Employees*

***ORDER BY*** *employee\_id;*



В данных случаях мы использовали аналитические функции чтобы упростить запросы и сделать их более понятными и комплексными (ибо как мы поняли, по сути любой запрос можно представить множеством соединений).

## *Выводы*

В ходе работы были изучены и проанализированы различные функции и методы запросов, приведены примеры их использования, а именно: агрегатные и иерархические функции, соединения и объединения, а так же механизмы подзапросов.