# Лабораторная работа по теме “Язык PL/pgSQL”

## Задача 1: Расчет стипендии для студентов

У вас есть две таблицы в базе данных: **students** и **activity\_scores**. Таблица **students** содержит информацию о студентах, их идентификаторах и общем балле. Таблица **activity\_scores** содержит информацию о баллах за разные виды деятельности для каждого студента.

Создайте таблицу **students** с колонками:

* **id** (SERIAL) - идентификатор студента (PRIMARY KEY)
* **name** (TEXT) - имя студента
* **total\_score** (INTEGER) - общий балл студента

Создайте таблицу **activity\_scores** с колонками:

* **student\_id** (INTEGER) - ссылка на студента в таблице students
* **activity\_type** (TEXT) - вид деятельности (например, "Homework", "Exam" и т.д.)
* **score** (INTEGER) - балл за деятельность

При наличии таблицы в базе, можете использовать существующую.

Создайте функцию **calculate\_scholarship**, которая будет рассчитывать стипендию для студента. Стипендия зависит от общего балла студента:

* Если общий балл больше или равен 90, стипендия равна 1000.
* Если общий балл больше или равен 80, но меньше 90, стипендия равна 500.
* В остальных случаях, стипендия равна 0.

Создайте триггер **update\_scholarship\_trigger**, который будет автоматически вызывать функцию **calculate\_scholarship** при обновлении баллов за деятельность в таблице **activity\_scores**. Протестируйте решение, вставив данные о студентах и их баллах за деятельность. Посмотрите, как автоматически обновляется стипендия каждого студента после добавления баллов.

## Задача 2: Учет баллов студентов

Представьте, что вы разрабатываете систему для учета баллов студентов в университете. Вам необходимо создать функциональность, которая автоматически будет обновлять общий балл каждого студента на основе полученных им баллов за разные виды деятельности.

Создайте таблицу **students**, содержащую следующие поля:

* **id** (SERIAL, PRIMARY KEY)
* **name** (TEXT)
* **total\_score** (INTEGER)

Создайте таблицу **activity\_scores**, содержащую следующие поля:

* **student\_id** (INTEGER, FOREIGN KEY к полю id таблицы students)
* **activity\_type** (TEXT)
* **score** (INTEGER)

При наличии таблицы в базе, можете использовать существующую.

Напишите функцию **update\_total\_score(student\_id INTEGER)**:

* Эта функция должна пересчитывать общий балл студента на основе баллов за разные виды деятельности в таблице **activity\_scores**.
* Используйте цикл для итерации по всем записям в **activity\_scores** для заданного **student\_id**.
* Обновите поле **total\_score** для соответствующего студента в таблице **students** суммой всех баллов за разные виды деятельности.

Напишите триггер, который будет автоматически вызывать функцию **update\_total\_score** при вставке новых записей в таблицу **activity\_scores**.

Предоставьте примеры использования:

* Вставьте несколько студентов в таблицу students.
* Вставьте записи о баллах за разные виды деятельности в таблицу activity\_scores.
* После вставки баллов, убедитесь, что общий балл каждого студента автоматически обновлен в таблице students.

# Решение:

[Ссылка на репозиторий](https://github.com/keeper0null/hwII/tree/main/homeWorks/hw10)

**create** **table** **if** **not** **exists** students

(

id serial

,**name** **text**

,total\_score **integer**

,**primary** **key**(id)

);

**comment** **on** **column** students.id **is** 'идентификатор студента';

**comment** **on** **column** students.**name** **is** 'имя студента';

**comment** **on** **column** students.total\_score **is** 'общий балл студента';

**create** **table** **if** **not** **exists** activity\_scores

(

student\_id **integer** **references** students (id)

,activity\_type **text**

,score **integer**

);

**comment** **on** **column** activity\_scores.student\_id **is** 'ссылка на студента в таблице students';

**comment** **on** **column** activity\_scores.activity\_type **is** 'вид деятельности';

**comment** **on** **column** activity\_scores.score **is** 'балл за деятельность';

**create** **or** **replace** **function** calculate\_scholarship(p\_total\_score **integer**)

**returns** **integer** **as** **$$**

**begin**

**return** **case**

**when** p\_total\_score >= 90 **then** 1000

**when** p\_total\_score >= 80 **and** p\_total\_score < 90 **then** 500

**else** 0

**end** **case**;

**end**;

**$$** **language** plpgsql

**alter** **table** students **add** **column** scholarship **integer**;

**comment** **on** **column** students.scholarship **is** 'стипендия студента';

**create** **or** **replace** **function** calculate\_total\_score(student\_id **integer**)

**returns** **integer** **as** **$$**

**declare**

p\_student\_id alias **for** **$1**;

v\_total\_score **integer** := 0;

c\_scores record;

**begin**

**for** c\_scores **in** (

**select** s.score

**from** activity\_scores s

**where** s.student\_id = p\_student\_id

) **loop**

v\_total\_score := v\_total\_score + c\_scores.score;

**end** **loop**;

**return** v\_total\_score;

**end**;

**$$** **language** plpgsql

**create** **or** **replace** **function** update\_total\_score()

**returns** **trigger** **as** **$$**

**declare**

v\_total\_score **integer**;

v\_scholarship **integer**;

**begin**

v\_total\_score := calculate\_total\_score(student\_id => **new**.student\_id);

v\_scholarship := calculate\_scholarship(p\_total\_score => v\_total\_score);

**update** students

**set** total\_score = v\_total\_score

,scholarship = v\_scholarship

**where** id = **new**.student\_id;

**return** **new**;

**end**;

**$$** **language** plpgsql

**create** **or** **replace** **trigger** update\_total\_score\_trigger

**after** **insert** **or** **update** **or** **delete** **on** activity\_scores

**for** **each** **row** **execute** **procedure** update\_total\_score();

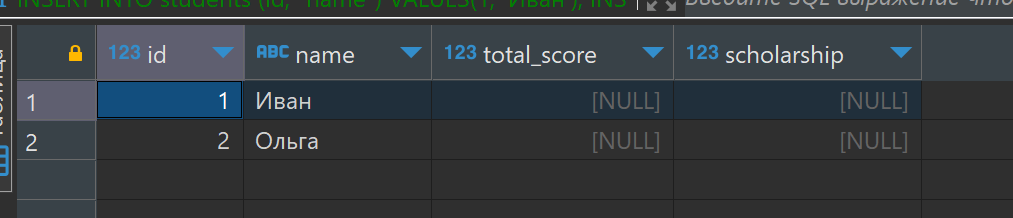
## Проверка работы:

**INSERT** **INTO** students (id, "name") **VALUES**(1, 'Иван');

**INSERT** **INTO** students (id, "name") **VALUES**(2, 'Ольга');

**select** \*

**from** students s



Вставка одной записи в activity\_scores

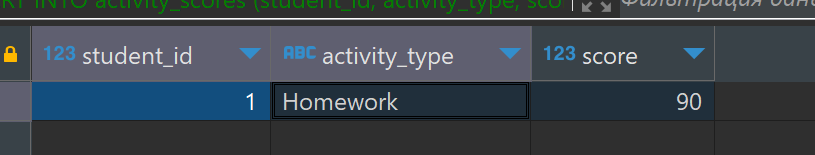
**INSERT** **INTO** activity\_scores

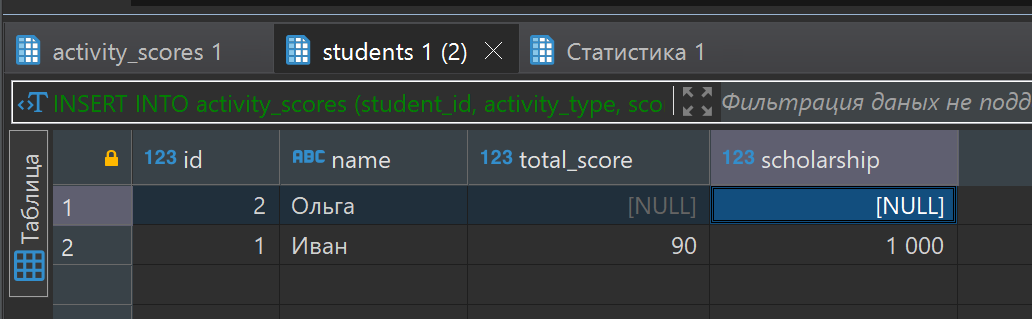
(student\_id, activity\_type, score)

**VALUES**(1, 'Homework', 90);

**select** \* **from** activity\_scores s;

**select** \* **from** students s;





Пересчитался итоговый бал и стипендия

Вставим еще две записи

**INSERT** **INTO** activity\_scores

(student\_id, activity\_type, score)

**VALUES**(2, 'Exam', 60);

**INSERT** **INTO** activity\_scores

(student\_id, activity\_type, score)

**VALUES**(1, 'Exam', 40);

**select** \* **from** activity\_scores s;

**select** \* **from** students s;

