| **№**  **задачи** | **Задание** | **Текст запроса** |
| --- | --- | --- |
|  | Создайте таблицу students с колонками:   * Id (SERIAL) – идентификатор студента (PRIMARYKEY) * 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   **//**Для видов активности создала отдельную таблицу.  А также для студентов задала значения по умолчанию для общего балла и для стипендии (насколько помню, сначала стипендия была у всех, до первой сессии) | **CREATE** **TABLE** students (  id serial **CONSTRAINT** students\_PK **PRIMARY** **KEY**,  **name** **varchar**(40) **NOT** **NULL**,  total\_score **integer** **default** 0,  scholarship **integer** **default** 1000  );  **INSERT** **INTO** students **values**  (**DEFAULT**, 'Sleepy'),  (**DEFAULT**, 'Doc'),  (**DEFAULT**, 'Bashful'),  (**DEFAULT**, 'Sneezy'),  (**DEFAULT**, 'Happy'),  (**DEFAULT**, 'Grumpy'),  (**DEFAULT**, 'Dopey');  **CREATE** **TABLE** activity\_types (  id serial **CONSTRAINT** activity\_PK **PRIMARY** **KEY**,  **name** **varchar**(40) **NOT** **NULL**  );  **INSERT** **INTO** activity\_types **values**  (**DEFAULT**, 'Homework'),  (**DEFAULT**, 'Colloquium') ,  (**DEFAULT**, 'Exam');  **CREATE** **TABLE** activity\_scores (  student\_id **integer** **REFERENCES** students (id),  activity\_type **integer** **REFERENCES** activity\_types (id),  score **integer**  ); |
| 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.  **//**Сделала через триггерную функцию, а также добавила в триггер и событие добавления записи | **CREATE** **OR** **REPLACE** **FUNCTION** calculate\_scholarship()  **RETURNS** **TRIGGER** **AS** **$$**  **DECLARE**  sum\_score **integer**;  scholarship\_value **integer**;  **BEGIN**  --Получаем и заполняем общий балл студента  **SELECT** **sum**(score) **INTO** sum\_score **FROM** activity\_scores  **WHERE** student\_id = **NEW**.student\_id;  **UPDATE** Students **SET** total\_score = sum\_score **where** id = **NEW**.student\_id;  --Вычисляем стипендию по общему баллу  **IF** sum\_score >= 90 **THEN**  scholarship\_value := 1000;  **ELSEIF** sum\_score >= 80 **AND** sum\_score < 90 **THEN**  scholarship\_value := 500;  **ELSE**  scholarship\_value := 0;  **END** **IF**;  **UPDATE** Students **SET** scholarship = scholarship\_value **where** id = **NEW**.student\_id;  **RETURN** **NEW**;  **RETURN** **NULL**;  **END**;  **$$** **LANGUAGE** plpgsql;  **CREATE** **OR** **REPLACE** **TRIGGER** update\_scholarship\_trigger  **AFTER** **INSERT OR** **UPDATE** **ON** activity\_scores  **FOR** **EACH** **ROW**  **EXECUTE** **FUNCTION** calculate\_scholarship (); |
| Протестируйте решение, вставив данные о студентах и их баллах за деятельность. Посмотрите, как автоматически обновляется стипендия каждого студента после добавления баллов.  **//**Очищала таблицу студентов, поэтому последовательность не с 1 | **Таблица activity\_scores изначально пуста**  **Начальный состав таблицы Students:**  **SELECT** \* **FROM** Students **LIMIT** 10    **Результат выполнения запросов**  **INSERT** **INTO** activity\_scores **values**  (8, 1, 20), (8, 2, 30), (8, 3, 49), (9, 1, 15), (9, 2, 30), (9, 3, 30), (10, 1, 0), (10, 2, 15), (10, 3, 20);  **SELECT** \* **FROM** activity\_scores    **SELECT** \* **FROM** students **ORDER BY** id    **Результат выполнения запросов**  **UPDATE** activity\_scores **SET** score = 0 **WHERE** student\_id = 8 **AND** activity\_type = 1;  **UPDATE** activity\_scores **SET** score = 50 **WHERE** student\_id = 9 **AND** activity\_type = 3;  **UPDATE** activity\_scores **SET** score = 20 **WHERE** student\_id = 10 **AND** activity\_type = 1;  **UPDATE** activity\_scores **SET** score = 50 **WHERE** student\_id = 10 **AND** activity\_type = 3;  **SELECT** \* **FROM** students **ORDER BY** id |
|  | 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 с колонками:   * Id (SERIAL) – идентификатор студента (PRIMARYKEY) * 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   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.  **//**Поскольку в первой задаче я уже автоматически в функции считаю общую сумму баллов студента при добавлении или редактировании записи, то создавать новую функцию и треггер не стала. Но чтобы выполнить задание по использованию цикла, переопределила функцию, включив в нее следующие изменения:  Для вычисления общего балла вместо sum(score) я использовала цикл:  sum\_score := 0;  **FOR** activity\_scores\_row **in** **select** \* **from** activity\_scores **where** student\_id = **NEW**.student\_id  **LOOP**  sum\_score := sum\_score + activity\_scores\_row.score;  **END** **LOOP**;  Для этого мне понадобилось добавить переменную, которая представляет собой строку таблицы activity\_scores:  activity\_scores\_row activity\_scores%ROWTYPE  Полный скрипт итоговой функции представлен справа | **CREATE** **OR** **REPLACE** **FUNCTION** calculate\_scholarship()  **RETURNS** **TRIGGER** **AS** **$$**  **DECLARE**  sum\_score **integer**;  scholarship\_value **integer**;  activity\_scores\_row activity\_scores%ROWTYPE;  **BEGIN**  --Вычисляем общий балл студента  sum\_score := 0;  **FOR** activity\_scores\_row **in** **select** \* **from** activity\_scores **where** student\_id = **NEW**.student\_id  **LOOP**  sum\_score := sum\_score + activity\_scores\_row.score;  **END** **LOOP**;  **UPDATE** Students **SET** total\_score = sum\_score **where** id = **NEW**.student\_id;  --Вычисляем стипендию по общему баллу  **IF** sum\_score >= 90 **THEN**  scholarship\_value := 1000;  **ELSEIF** sum\_score >= 80 **AND** sum\_score < 90 **THEN**  scholarship\_value := 500;  **ELSE**  scholarship\_value := 0;  **END** **IF**;  **UPDATE** Students **SET** scholarship = scholarship\_value **where** id = **NEW**.student\_id;  **RETURN** **NEW**;  **RETURN** **NULL**;  **END**;  **$$** **LANGUAGE** plpgsql; |
| 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. | **Результат выполнения запросов для начальных данных:**  **SELECT** \* **FROM** activity\_scores    **SELECT** \* **FROM** Students **LIMIT** 10    **Результат выполнения запросов**  **INSERT** **INTO** activity\_scores **values**  (11, 1, 20), (11, 2, 30), (11, 3, 50), (12, 1, 15), (12, 2, 15), (12, 3, 15), (13, 1, 20), (13, 2, 30), (13, 3, 31);  **SELECT** \* **FROM** activity\_scores    **SELECT** \* **FROM** students **ORDER BY** id |