федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет инфокоммуникационных технологий

Лабораторная работа №3

«Процедуры, функции, триггеры в PostgreSQL»

по дисциплине: «Проектирование и реализация баз данных»

Выполнила:

студентка III курса ИКТ группы К33402 Самчук Анита Алексеевна

Проверила:

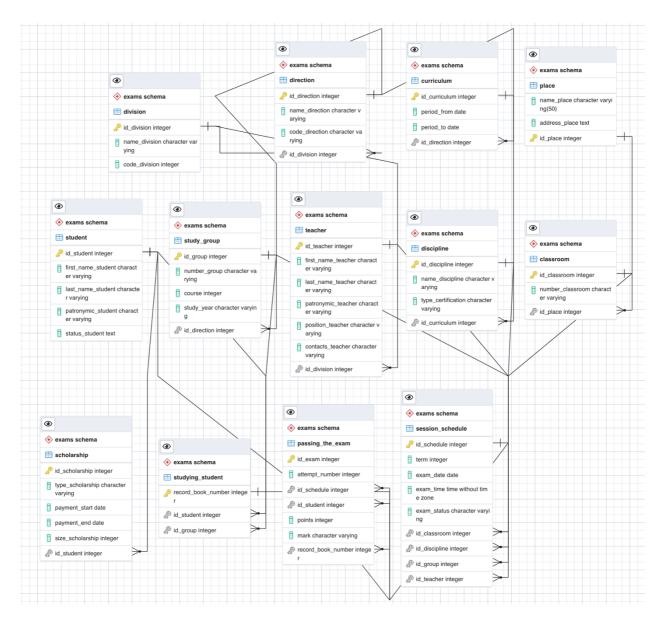
Говорова Марина Михайловна

Цель работы: овладеть практическими создания и использования процедур, функций и триггеров в базе данных PostgreSQL.

Практическое задание:

- 1. Создать процедуры/функции согласно индивидуальному заданию часть 4.
- 2. Создать триггер для логирования событий вставки, удаления, редактирования данных в базе данных PostgreSQL (согласно индивидуальному заданию, часть 5). Допустимо создать универсальный триггер или отдельные триггеры на логирование действий.

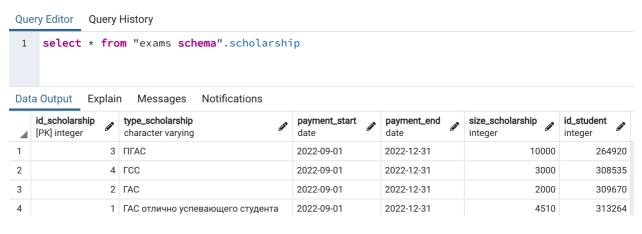
Схема базы данных:



Задание 1. Создайте хранимые процедуры

1. Для повышения стипендии отличникам на 10%

До:



Создание:

```
Query Editor
            Query History
    create procedure "exams schema".raise_scholarship()
1
 2
    language sql
   as $$
 3
   update "exams schema".scholarship
 4
    set size_scholarship = size_scholarship * 1.1 where id_scholarship in
   (select id_scholarship from "exams schema".scholarship
 6
    where type_scholarship = 'ГАС отлично успевающего студента' and
 7
 8
    current_timestamp between payment_start and payment_end);
 9
    $$;
Data Output
           Explain
                    Messages
                              Notifications
CREATE PROCEDURE
Query returned successfully in 32 msec.
```

После:

Query Editor Query History 1 call "exams schema".raise_scholarship(); 2 select * from "exams schema".scholarship;

Data	a Output Explair	n Messages Notifications				
	id_scholarship [PK] integer	type_scholarship character varying	payment_start date	payment_end date	size_scholarship integer	id_student integer
1	3	ПГАС	2022-09-01	2022-12-31	10000	264920
2	4	ГСС	2022-09-01	2022-12-31	3000	308535
3	2	ГАС	2022-09-01	2022-12-31	2000	309670
4	1	ГАС отлично успевающего студента	2022-09-01	2022-12-31	4961	313264

2. Для перевода студентов на следующий курс

До:

```
SELECT * FROM "exams schema".study_group
 2
     ORDER BY id_group ASC
                                      Notifications
Data Output
              Explain
                         Messages
    id_group
                   number_group
                                                                        id_direction
                                        course
                                                   study_year
   [PK] integer
                   character varying
                                                   character varying
                                                                        integer
                                        integer
                1 K33401
                                                 3 2022/2023
1
                                                                                     1
                2 K3243
                                                 2 2022/2023
                                                                                     2
2
3
                3 M3112
                                                 1
                                                   2022/2023
                                                                                     3
                4 P34684
4
                                                  2022/2023
                                                                                     4
5
                5 K4142
                                                 1 2022/2023
                                                                                     1
```

После:

5 K4242

2 2023/2024

```
Query Editor Query History
create procedure "exams schema".update_course()
 2 language sql
3 as $$
 4 update "exams schema".study_group
 5 set course = course + 1,
        study_year = concat(cast(substring(study_year, 1, 4) as int)+1 , '/', cast(substring(study_year, 6, 4) as int) + 1),
        number_group = concat(substring(number_group, 1, 2),
                               cast(substring(number_group, 3, 1) as int) + 1,
                               substring(number_group, 4, 4))
10 where course < 4;
11 $$;
12
13 call "exams schema".update_course();
14 select * from "exams schema".study_group;
Data Output Explain Messages Notifications
                                                         id_direction
   id_group
[PK] integer

number_group
character varying

course integer
integer

study_year
character varying
             4 P34684
                                       4 2022/2023
                                                                    4
2
             1 K34401
                                       4 2023/2024
             2 K3343
                                       3 2023/2024
                                                                    2
             3 M3212
                                       2 2023/2024
                                                                    3
```

3. Для изменения оценки при успешной пересдаче экзамена

До:

Query Editor Query History

```
1 SELECT * FROM "exams schema".passing_the_exam
2 ORDER BY id_exam ASC
```

Dat	ta Output Exp	lain Messages	Notifications				
4	id_exam [PK] integer	attempt_number integer	id_schedule integer	id_student integer	points integer	mark character varying	record_book_number integer
1	1	1	1	313264	20	5	5
2	2	1	2	309670	20	5	4
3	3	1	4	313264	20	5	5
4	4	1	3	282928	0	Незачет	2
5	5	1	5	264920	16	Зачет	1
6	6	1	6	247891	12	Зачет	8
7	7	1	4	282928	0	2	2
8	8	1	2	313264	18	4	5
9	9	1	3	264920	16	Зачет	1
10	10	1	1	309670	20	5	4
11	11	1	1	282928	0	2	2
12	12	3	7	282928	0	2	2
13	13	1	3	308535	20	Зачет	3

После:

Query Editor Query History

```
1 create procedure "exams schema".retake(schedule_id integer, points integer, mark integer, number_record_book integer)
  2 language sql
   3 as $$
   4 insert into "exams schema".passing_the_exam(id_exam, attempt_number, id_student, id_schedule, points, mark, record_book_numbe
               values((select max(id_exam) from "exams schema".passing_the_exam) + 1,
                                             (select max(attempt_number) from "exams schema".passing_the_exam
                                             where id_schedule = schedule_id and record_book_number = number_record_book) + 1,
                                             (select id_student from "exams schema".studying_student
   8
                                             \begin{tabular}{ll} \textbf{where} & \texttt{record\_book\_number} & = & \texttt{number\_record\_book}) \end{tabular}, \\ \begin{tabular}{ll} \textbf{vertical pook} & \texttt{vertical pook} & \texttt{vert
   9
10
                                             schedule_id, points, mark, number_record_book);
11 $$;
12
13 call "exams schema".retake(4, 14, 3, 2);
14 select * from "exams schema".passing_the_exam;
```

	id_exam [PK] integer	attempt_number integer	id_schedule integer	id_student integer	points integer	mark character varying	record_book_number_integer
1	1		1 1	313264			5
2	2		1 2	309670	20	5	4
3	3		1 4	313264	20	5	5
4	4		1 3	282928	0	Незачет	2
5	5		1 5	264920	16	Зачет	1
6	6		1 6	247891	12	Зачет	8
7	7		1	282928	0	2	2
8	8		1 2	313264	18	4	5
9	9		1 3	264920	16	Зачет	1
10	10		1 1	309670	20	5	4
11	11		1 1	282928	0	2	2
12	12		3 7	282928	0	2	2
13	13		1 3	308535	20	Зачет	3
14	14		2 4	282928	14	3	2

```
Query Editor Query History
   create or replace function "exams schema".add_to_log() returns trigger as $$
3
        mstr varchar(30);
4
        astr varchar(100);
        retstr varchar(254);
5
6 ▼ begin
        if TG_OP = 'INSERT' then
7 ▼
8
            astr = new;
            mstr := 'Add data';
9
10
            retstr := mstr || astr;
            insert into "exams schema".logs(text, added, table_name)
11
            values(retstr, now(), TG_TABLE_NAME());
12
13
            return new;
        elsif TG_OP = 'UPDATE' then
14
            astr = new;
15
            mstr := 'Update data';
16
            retstr := mstr || astr;
17
            insert into "exams schema".logs(text, added, table_name)
18
            values(retstr, now(), TG_TABLE_NAME());
19
            return new;
20
        elsif TG_OP = 'DELETE' then
21
22
            astr = old;
23
            mstr := 'Remove data';
24
            retstr := mstr || astr;
            insert into "exams schema".logs(text, added, table_name)
25
            values(retstr, now(), TG_TABLE_NAME());
26
27
            return old;
        end if;
28
29
    end:
30 $$ language nlngsql:
Data Output Explain
                             Notifications
                  Messages
CREATE FUNCTION
```

Query returned successfully in 56 msec.

Проблема:

```
Query Editor Query History
insert into "exams schema".passing_the_exam(
2 id_exam, attempt_number, id_schedule, id_student, points, mark, record_book_number)
   values (15, 1, 3, 282928, 12, '3aчет', 2);
3
   update "exams schema".passing_the_exam
5
   set id_exam = 16
6
7
   where id_exam = 15;
8
9
   delete from "exams schema".passing_the_exam
   where id_exam = 16;
10
11
12 select * from "exams schema".logs
Data Output Explain Messages Notifications
ERROR: relation "exams schema.logs" does not exist
LINE 1: insert into "exams schema".logs(text, added, table_name)
QUERY: insert into "exams schema".logs(text, added, table_name)
              values(retstr, now(), TG_TABLE_NAME())
CONTEXT: PL/pgSQL function "exams schema".add_to_log() line 11 at SQL statement
SQL state: 42P01
```