

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
Факультет инфокоммуникационных технологий

ОТЧЕТ
О ЛАБОРАТОРНОЙ РАБОТЕ № 3
по теме: процедуры, функции, триггеры в PostgreSQL.
по дисциплине: Проектирование и реализация баз данных

Специальность:

09.03.03 Мобильные и сетевые технологии

Проверил:

Говорова М.М. _____

Дата: «__» _____ 2021 г.

Оценка _____

Выполнил:

студентка группы К3240

Бабан Виктория

Санкт-Петербург 2022 г.

ЦЕЛЬ РАБОТЫ

Овладеть практическими создания и использования процедур, функций и триггеров в базе данных PostgreSQL.

Оборудование: компьютерный класс.

Программное обеспечение: СУБД PostgreSQL, SQL Shell (psql).

Вариант 1. Практическое задание:

- I. Создать процедуры/функции согласно индивидуальному заданию и (согласно индивидуальному заданию, часть 4).
- II. 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



Выполнение:

Задание 1. Создайте хранимые процедуры.

1) Для повышения стипендии отличникам на 10%.

```
1 CREATE OR REPLACE PROCEDURE "SESSION".raise_scholarship()
2 LANGUAGE plpgsql AS
3 $$
4 BEGIN
5 UPDATE "SESSION".scholarship
6 SET size_scholarship = size_scholarship * 1.1, type_scholarship = 'Повышенная',
7   payment_start_date= (CASE WHEN date_part('month', payment_start_date) = 2
8   THEN payment_start_date + INTERVAL '5 months'
9   ELSE payment_start_date + INTERVAL '7 months' END),
10  payment_end_date= (CASE WHEN date_part('month', payment_end_date) = 1
11  THEN payment_end_date + INTERVAL '5 months'
12  ELSE payment_end_date + INTERVAL '7 months 1 day' END)
13 WHERE id_scholarship IN (SELECT id_scholarship FROM "SESSION".scholarship
14   WHERE number_record_book IN (SELECT DISTINCT number_record_book FROM "SESSION".passing_exam AS q
15   WHERE (SELECT AVG(CAST(score AS int))
16   FROM "SESSION".passing_exam AS b
17   WHERE q.number_record_book = b.number_record_book
18   AND (score NOT IN ('Зачёт', 'Незачёт')) = 5)
19   AND CURRENT_TIMESTAMP BETWEEN payment_start_date AND payment_end_date);
20 END;
21 $$;
```

План выполнения

Результат

Сообщения

Notifications

CREATE PROCEDURE

Запрос завершён успешно, время выполнения: 119 msec.

До:

	id_scholarship [PK] integer	type_scholarship character varying (20)	payment_start_date date	payment_end_date date	size_scholarship integer	number_record_book integer
1	1	Повышенная	2022-02-01	2022-06-30	4100	312310
2	2	Повышенная	2022-02-01	2022-06-30	6000	283128
3	3	Базовая	2022-02-01	2022-06-30	2000	312407
4	4	Базовая	2022-02-01	2022-06-30	2000	336100
5	5	Социальная	2022-02-01	2022-06-30	3000	283991

После:

```
1 CALL "SESSION".raise_scholarship();
2 SELECT * FROM "SESSION".scholarship
```

План выполнения

Результат

Сообщения

Notifications

	id_scholarship [PK] integer	type_scholarship character varying (20)	payment_start_date date	payment_end_date date	size_scholarship integer	number_record_book integer
1	2	Повышенная	2022-02-01	2022-06-30	6000	283128
2	4	Базовая	2022-02-01	2022-06-30	2000	336100
3	5	Социальная	2022-02-01	2022-06-30	3000	283991
4	1	Повышенная	2022-07-01	2023-01-31	4510	312310
5	3	Повышенная	2022-07-01	2023-01-31	2200	312407

2) Для перевода студентов на следующий курс.

```
1 CREATE PROCEDURE "SESSION".update_course()
2 LANGUAGE plpgsql AS
3 $$
4 BEGIN
5 UPDATE "SESSION".group
6 SET course = course + 1,
7     group_number = CONCAT(SUBSTRING(group_number, 1, 2),
8                           CAST(SUBSTRING(group_number, 3, 1) AS INT) + 1,
9                           SUBSTRING(group_number, 4, 3)),
10    study_year = CONCAT(CAST(SUBSTRING(study_year, 1, 4) AS INT) + 1, '/',
11                        CAST(SUBSTRING(study_year, 6, 4) AS INT) + 1)
12 WHERE CAST(SUBSTRING(group_number, 3, 1) AS INT) != 4;
13 END;
14 $$;
```

План выполнения Результат Сообщения Notifications

CREATE PROCEDURE

Запрос завершён успешно, время выполнения: 87 msec.

До:

	id_group [PK] integer	group_number character varying (10)	course integer	id_direction integer	study_year character varying (10)	
1	1	K3240	2	3	2021/2022	
2	2	K3140	1	3	2021/2022	
3	3	N33481	4	4	2021/2022	
4	4	K3442	4	1	2021/2022	
5	5	K3220	2	2	2021/2022	
6	6	K3121	1	2	2021/2022	
7	7	K3143	1	1	2021/2022	

После:

```

1 CALL "SESSION".update_course();
2 SELECT * FROM "SESSION"."group"
3 ORDER BY id_group ASC

```

План выполнения Результат Сообщения Notifications

	id_group [PK] integer	group_number character varying (10)	course integer	id_direction integer	study_year character varying (10)
1	1	K3340	3	3	2022/2023
2	2	K3240	2	3	2022/2023
3	3	N34481	5	4	2022/2023
4	4	K3442	4	1	2021/2022
5	5	K3320	3	2	2022/2023
6	6	K3221	2	2	2022/2023
7	7	K3243	2	1	2022/2023

3) Для изменения оценки при успешной пересдаче экзамена.

```

1 CREATE OR REPLACE PROCEDURE "SESSION".retake(shedule_id INTEGER, number_rec_book INTEGER, new_score VARCHAR(10))
2 LANGUAGE plpgsql AS
3 $$
4 BEGIN
5 INSERT INTO "SESSION".passing_exam(
6     id_shedule, attempt_number, number_record_book, score)
7 VALUES (shedule_id,
8     (SELECT MAX(attempt_number) FROM "SESSION".passing_exam
9         WHERE id_shedule = shedule_id AND number_record_book = number_rec_book) + 1,
10     number_rec_book,
11     new_score);
12 END;
13 $$;
14

```

План выполнения Результат Сообщения Notifications

CREATE PROCEDURE

Запрос завершён успешно, время выполнения: 160 мсек.

До:

	id_exam [PK] integer	id_shedule integer	attempt_number integer	number_record_book integer	score character varying (10)
1	1	1	1	312310	Зачёт
2	2	1	1	312407	Зачёт
3	4	10	1	336100	2
4	5	10	2	336100	4
5	6	9	1	312310	5
6	7	9	1	312407	5
7	8	7	1	283128	4
8	9	8	1	283128	3
9	10	7	1	283129	2
10	11	7	2	283129	2
11	12	7	3	283129	2
12	14	7	1	283991	5
13	15	8	1	283991	4
14	17	1	1	312539	Незачёт

После:

```

1 CALL "SESSION".retake(1, 312539, 'Зачёт');
2 SELECT * FROM "SESSION".passing_exam
3 ORDER BY id_exam ASC

```

План выполнения	Результат	Сообщения	Notifications		
	<div>id_exam</div> <div>[PK] integer</div>	<div>id_shedule</div> <div>integer</div>	<div>attempt_number</div> <div>integer</div>	<div>number_record_book</div> <div>integer</div>	<div>score</div> <div>character varying (10)</div>
1	1	1	1	312310	Зачёт
2	2	1	1	312407	Зачёт
3	4	10	1	336100	2
4	5	10	2	336100	4
5	6	9	1	312310	5
6	7	9	1	312407	5
7	8	7	1	283128	4
8	9	8	1	283128	3
9	10	7	1	283129	2
10	11	7	2	283129	2
11	12	7	3	283129	2
12	14	7	1	283991	5
13	15	8	1	283991	4
14	17	1	1	312539	Незачёт
15	18	1	2	312539	Зачёт

Задание 2. Создать триггер для логирования событий вставки, удаления, редактирования

```
1 CREATE OR REPLACE FUNCTION "SESSION".add_to_log() RETURNS TRIGGER AS $$
2 DECLARE
3     mstr varchar(30);
4     astr varchar(100);
5     retstr varchar(254);
6 BEGIN
7     IF TG_OP = 'INSERT' THEN
8         astr = NEW;
9         mstr := 'ADD DATA';
10        retstr := mstr||astr;
11        INSERT INTO "SESSION".logs(text, added, table_name)
12            values (retstr, NOW(), TG_TABLE_NAME());
13        RETURN NEW;
14    ELSIF TG_OP = 'UPDATE' THEN
15        astr = NEW;
16        mstr := 'UPDATE DATA';
17        retstr := mstr||astr;
18        INSERT INTO "SESSION".logs(text, added, table_name)
19            values (retstr, NOW(), TG_TABLE_NAME());
20        RETURN NEW;
21    ELSIF TG_OP = 'DELETE' THEN
22        astr = OLD;
23        mstr := 'REMOVE DATA';
24        retstr := mstr || astr;
25        INSERT INTO "SESSION".logs(text, added, table_name)
26            values (retstr, NOW(), TG_TABLE_NAME());
27        RETURN OLD;
28    END IF;
29 END;
30 $$ LANGUAGE plpgsql;
```

План выполнения Результат Сообщения Notifications

CREATE FUNCTION

Запрос завершён успешно, время выполнения: 323 msec.

```
1 CREATE TRIGGER t_pass_exam AFTER INSERT OR UPDATE OR DELETE ON "SESSION".passing_exam
2 FOR EACH ROW EXECUTE PROCEDURE "SESSION".add_to_log();
```

План выполнения Результат Сообщения Notifications

CREATE TRIGGER

Запрос завершён успешно, время выполнения: 97 msec.


```

1 INSERT INTO "SESSION".passing_exam(
2     id_schedule, attempt_number, number_record_book, score)
3     VALUES (1, 1, 283991, 'Зачёт');

```

План выполнения Результат Сообщения Notifications

INSERT 0 1

Запрос завершён успешно, время выполнения: 85 msec.

```

1 UPDATE "SESSION".passing_exam
2     SET score='Незачёт'
3     WHERE id_schedule = 1 AND number_record_book = 283991;

```

План выполнения Результат Сообщения Notifications

UPDATE 1

Запрос завершён успешно, время выполнения: 112 msec.

```

1 DELETE FROM "SESSION".passing_exam
2     WHERE id_exam = 21;

```

План выполнения Результат Сообщения Notifications

DELETE 1

Запрос завершён успешно, время выполнения: 81 msec.

```

1 SELECT * FROM "SESSION".logs
2

```

План выполнения Результат Сообщения Notifications

	text	added
	text	timestamp without time zone
1	ADD DATA(21,1,1,283991,Зачёт)	2022-05-26 18:49:29.105999
2	UPDATE DATA(21,1,1,283991,Незачёт)	2022-05-26 18:51:05.007588
3	REMOVE DATA(21,1,1,283991,Незачёт)	2022-05-26 18:52:28.697483

ВЫВОДЫ

В данной работе я изучила функции и процедуры, создала их под разные задачи и условия. Также я узнала многое про триггеры и использовала их для хранения информации об изменениях в данных своей базы