Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по лабораторной работе №5
«Процедуры, функции, триггеры в PostgreSQL»

по дисциплине **«Проектирование и реализация баз данных»**

Автор: Пеньков Г.Д.

Факультет: ИКТ

Группа: К3241

Преподаватель: Говорова М. М.

Санкт-Петербург 2023

Цель работы

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

Вариант 3. БД «Библиотека»

Практическое задание:

Вариант 2 (max - 8 баллов)

1. Создать процедуры/функции согласно индивидуальному заданию (часть 4).
2. Модифицировать триггер (триггерную функцию) на проверку корректности входа и выхода сотрудника (см. Практическое задание 1 Лабораторного практикума (Приложение)) с максимальным учетом «узких» мест некорректных данных по входу и выходу).
3. Создать авторский триггер по варианту индивидуального задания.

Выполнение

1. Создать процедуры/функции согласно индивидуальному заданию (часть 4).

   1.1. Для проверки наличия экземпляров заданной книги в библиотеке (процедура должна возвращать количество экземпляров книги).

```
create or replace function lab3.check_book_availability(_book_id int)
  returns int
  language plpgsql
as
$$
  declare num_of_available_copies int;
  begin
    if _book_id not in (select book_id from lab3.books) then
    raise exception 'Книга с id % не найдена', _book_id;
  end if;
  SELECT COUNT(*) INTO NUM_OF_AVAILABLE_COPIES
    FROM LAB3.PUBLICATION_COPIES AS PC
    JOIN
      (SELECT PC.INVENTORY_NUMBER
      FROM LAB3.PUBLICATION_COPIES AS PC
      WHERE PC.WRITE_OFF_ACT IS NULL
      EXCEPT SELECT BOOK
      FROM LAB3.BOOK_LENDING AS BL
      WHERE REAL_RETURN_DATE IS NULL) AS AVAILABLE_COPIES
    ON AVAILABLE_COPIES.INVENTORY_NUMBER = PC.INVENTORY_NUMBER
    JOIN LAB3.PUBLICATIONS AS PUB ON PC.ISBN = PUB.ISBN
    JOIN LAB3.BOOKS AS B ON B.BOOK_ID = PUB.BOOK
    WHERE B.BOOK_ID = _book_id;
    return NUM_OF_AVAILABLE_COPIES;
  end;
$$;
select * from lab3.check_book_availability(1)
```

Data Output | Messages | Notifications

	check_book_availability integer
1	4

1.2. Для ввода в базу данных новой книги.

```sql
CREATE OR REPLACE PROCEDURE lab3.add_book(
  title VARCHAR(100),
  original_lang_code varchar(2),
  author varchar(50),
  year numeric(4) default null,
  bbk varchar(50) default null,
  udk varchar(50) default null
  )
language plpgsql
as $$
declare
  _author_id int;
  _book_id int;
begin
  if author not in (select name from lab3.authors) then
      insert into lab3.authors (name) values (author) returning author_id into _author_id;
  else
    select author_id into _author_id from lab3.authors where name = author;
    end if;

  if original_lang_code not in (select lang_code from lab3.book_original_languages) then
    insert into lab3.book_original_languages (lang_code) values (original_lang_code);
    end if;

  insert into lab3.books (title, year, bbk, udc, original_language)
  values (title, year, bbk, udk, original_lang_code) returning book_id into _book_id;

  insert into lab3.authorships (author, book) values (_author_id, _book_id);
end;
$$;

```

```sql
call lab3.add_book(
    'Чайка по имени Джонатан Ливингстон',
    'en',
    'Ричард Бах',
    1970
);
```

```sql
select * from lab3.books order by book_id desc;
```

Data Output | Messages | Notifications

	book_id [PK] integer	title character varying (100)	year numeric (4)	bbk character varying (50)	udc character varying (50)	original_language character
1	19	Чайка по имени Джонатан Ливингстон	1970	[null]	[null]	en
2	18	Параллельное программирование на современном языке C++	2021	32.973.202	004.4	en
3	17	C# 20 в деталях	2020	32.372	004.42	en

```sql
select * from lab3.authors order by author_id desc;
```

Data Output | Messages | Notifications

	author_id [PK] integer	name character varying (50)
1	21	Ричард Бах
2	19	Райнер Гримм

```sql
select * from lab3.authorships order by authorship_id desc;
```

Data Output | Messages | Notifications

	authorship_id [PK] integer	author integer	book integer	order_num integer
1	28	21	19	[null]
2	27	16	14	[null]
3	26	16	13	[null]

1.3. Для ввода нового читателя (необходимо проверить наличие читателя в картотеке, чтобы не назначить ему номер вторично).

```sql
create or replace procedure lab3.add_reader_card(
    _name varchar(50),
    _passport_info varchar(200),
    _address varchar(100),
    _phone_num varchar(20) default null,
    _email varchar(50) default null,
    _education varchar(50) default null
    )
language plpgsql
as $$
begin
    if _passport_info in (select passport_info from lab3.reader_cards) then
        raise exception 'Данный читатель уже существует';
    end if;
    insert into lab3.reader_cards (passport_info, reader_name, phone_number, address, email, education)
    values (_passport_info, _name, _phone_num, _address, _email, _education);
end; $$;
```

```sql
call lab3.add_reader_card('Куприянов Александр Александрович',
                          '60 13 099279, выдан  ГУ МВД России по г. Саратову, кп 088-049',
                         'Санкт-Петербург, улица Ленина, дом 10')
```

Data Output | Messages | Notifications

```
ERROR:  Данный читатель уже существует
CONTEXT:  функция PL/pgSQL lab3.add_reader_card(character varying,character varying,character varying,character
RAISE

ОШИБКА:  Данный читатель уже существует
SQL state: P0001
```

```sql
call lab3.add_reader_card('Горохов Иван Дмитриевич',
                          '61 14 123279, выдан  ГУ МВД России по г. Воронеж, кп 089-087',
                         'Санкт-Петербург, улица Ленина, дом 12')
select * from lab3.reader_cards order by card_id desc;
```

Data Output | Messages | Notifications

	card_id [PK] integer	passport_info character varying (200)	reader_name character varying (50)
1	30	61 14 123279, выдан ГУ МВД России по г. Воронеж, кп 089-087	Горохов Иван Дмитриевич
2	29	63 24 620360, выдан ГУ МВД России по г. Санкт-Петербургу, кп 364-8...	Скворцов Евгений Александрович
3	28	85 10 671342, выдан ГУ МВД России по г. Уфе, кп 403-594	Гусева Ольга Вадимовна

2. Модификация триггера из практического задания

Для создания триггерной функции мы создали вспомогательную функцию, которая возвращает последнее время выхода (или входа) для заданного работника:

```
create or replace function get_last_punch(_employee_id int)
returns table (EMPloyee_ID int, IS_OUT_PUNCH bool, last_punch_time timestamp)
language plpgsql
as $$
    begin
        return query
            SELECT T.EMPLOYEE_ID, T.IS_OUT_PUNCH, T.PUNCH_TIME
            FROM (SELECT TP.EMPLOYEE_ID, MAX(TP.PUNCH_TIME) AS LATEST_TIME
                    FROM TIME_PUNCH TP
                    GROUP BY TP.EMPLOYEE_ID
                    HAVING TP.EMPLOYEE_ID = _employee_id
                ) LATEST_TIMES
            JOIN TIME_PUNCH T
            ON T.EMPLOYEE_ID = LATEST_TIMES.EMPLOYEE_ID AND T.PUNCH_TIME = LATEST_TIMES.LATEST_TIME;
    end;
$$;
select * from get_last_punch(1)
```

Data Output | Messages | Notifications

	employee_id integer	is_out_punch boolean	last_punch_time timestamp without time zone
1	1	true	2023-11-24 14:25:59.656886

Триггерная функция и создание триггера:

```
create or replace function fn_check_time_punch() returns trigger as $psql$
    declare
        _last_punch_time timestamp;
        _last_is_out_punch bool;
    begin
        select is_out_punch into _last_is_out_punch from get_last_punch(new.employee_id);
        select last_punch_time into _last_punch_time from get_last_punch(new.employee_id);

        -- нельзя выйти/войти 2 раза подряд
        if new.is_out_punch = _last_is_out_punch then return null;
        end if;
        -- выход не может быть первой записью (сначала нужно войти)
        if new.is_out_punch = true and _last_is_out_punch is null then return null;
        end if;
        -- новый выход/вход должен быть позже последнего
        if new.punch_time < _last_punch_time then return null;
        end if;
        return new;
    end;
$psql$ language plpgsql;

create trigger check_time_punch before insert on time_punch
for each row execute procedure fn_check_time_punch();
```

Примеры выполнения:

```
1  INSERT INTO TIME_PUNCH (EMPLOYEE_ID, IS_OUT_PUNCH)
2  VALUES (3, true);
3
4  select * from TIME_PUNCH order by id desc;
```

Data Output | Messages | Notifications

	id [PK] integer	employee_id integer	is_out_punch boolean	punch_time timestamp without time zone
1	26	2	true	2024-11-24 17:42:48
2	25	2	false	2023-11-24 14:42:54.730079
3	24	2	true	2023-11-24 14:28:12.761426

```
1  INSERT INTO TIME_PUNCH (EMPLOYEE_ID, IS_OUT_PUNCH)
2  VALUES (3, false);
3
4  select * from TIME_PUNCH order by id desc;
```

Data Output | Messages | Notifications

	id [PK] integer	employee_id integer	is_out_punch boolean	punch_time timestamp without time zone
1	30	3	false	2023-12-27 11:28:33.423699
2	26	2	true	2024-11-24 17:42:48
3	25	2	false	2023-11-24 14:42:54.730079

```
1  INSERT INTO TIME_PUNCH (EMPLOYEE_ID, IS_OUT_PUNCH, punch_time)
2  VALUES (3, false, '2022-01-01');
3
4  select * from TIME_PUNCH order by id desc;
```

Data Output | Messages | Notifications

	id [PK] integer	employee_id integer	is_out_punch boolean	punch_time timestamp without time zone
1	36	3	true	2023-12-27 11:31:16.284443
2	30	3	false	2023-12-27 11:28:33.423699

3. Создание триггера по варианту индивидуального задания

```
CREATE OR REPLACE FUNCTION FN_CHECK_BOOK_LENDING() RETURNS TRIGGER LANGUAGE PLPGSQL AS $$
begin
    -- нельзя выдать книгу, которая уже выдана
    if new.book in
        (select book from lab3.book_lending where real_return_date is null) then
            return null;
    end if;

    -- нельзя выдать книгу, которая списана с учета
    if new.book in
        (select inventory_number from lab3.publication_copies where write_off_act is not null) then
            return null;
    end if;

    -- нельзя выдать книгу, если у человека более чем 3 задолженных книги
    if (select num_of_fined_books from lab3.debtors where card_id = new.reader_card) > 3 then
        return null;
    end if;
    return new;
end; $$;

create trigger check_book_lending before insert on lab3.book_lending
for each row execute procedure fn_check_book_lending();
```

Примеры выполнения:

```
insert into lab3.book_lending (reader_card, book, employee)
values (24, 89, 4)
-- книга с инвентарным номером 89 уже взята
```

Data Output | Messages | Notifications

```
INSERT 0 0
```

```
insert into lab3.book_lending (reader_card, book, employee)
values (24, 73, 4)
-- книга с инвентарным номером 73 снята с учета
```

Data Output | Messages | Notifications

```
INSERT 0 0
```

```
1  insert into lab3.book_lending (reader_card, book, employee)
2  values (24, 172, 4)
3
4  select * from lab3.book_lending order by lending_id desc
```

Data Output | Messages | Notifications

	lending_id [PK] integer	reader_card integer	book integer	lend_date date	planned_return_date date	real_return_date date	return_info character varying (100)	employee integer	fine integer
1	70	24	172	2023-12-27	2024-01-27	[null]	[null]	4	0
2	67	24	95	2023-12-19	2023-12-30	[null]	[null]	4	0

```
1  insert into lab3.book_lending (reader_card, book, employee)
2  values (19, 160, 4)
3
```

Data Output | Messages | Notifications

```
INSERT 0 0
```

```
4  select * from lab3.debtors where card_id = 19
```

Data Output | Messages | Notifications

	card_id integer	reader_name character varying (50)	total_fine bigint	num_of_fined_books bigint
1	19	Калинина Тамара Ивановна	2255	6

Вывод

В данной лабораторной работе мы написали несколько процедур, функций и триггеров. Тем самым, мы познакомились с основами процедурного языка PL/pgSQL.