

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГАОУ ВО «СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт информационных технологий

ОТЧЕТ

о выполнении лабораторной работы №2
«Язык SQL. Генераторы. Функции. Триггеры»
по дисциплине
«Управление данными»

Выполнил:

ст.гр. ИС/б-23-1-о

Баймурадов Т. Р.

Севастополь, 2025

1. Цель

Выработать у обучающихся практические навыки по работе с реляционными базами данных, ознакомить с принципом работы генераторов, функций и триггеров.

2. Постановка задачи

1. В соответствии с вариантом задания (приложение А) создать генератор для каждого первичного ключа (исключение, если первичный ключ является символьным полем).

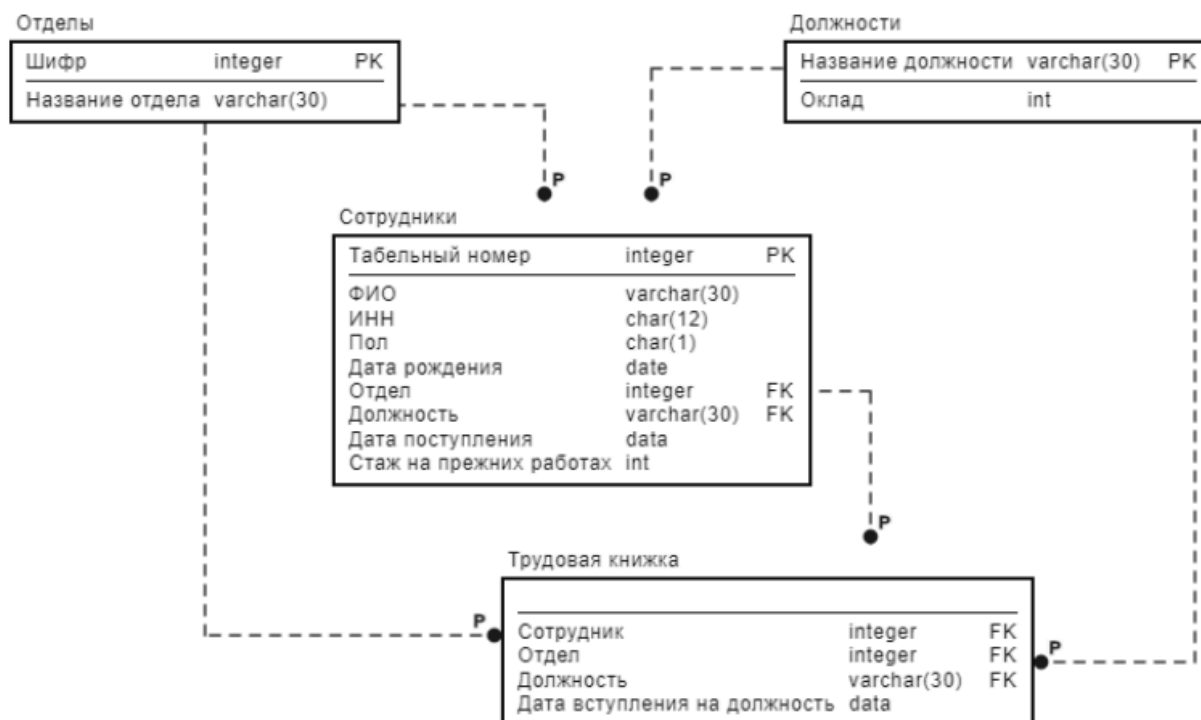
2. Согласно варианту, создать функции.

3. Согласно варианту, написать триггеры.

4. Написать отчет.

3. Вариант

Вариант №3.



Пол – значения – «м» и «ж», по умолчанию – «ж»

Рисунок 1 – Схема БД

Создание функций.

1. Функция, определяющая, является ли человек на текущую дату пенсионером. Параметры: пол и дата рождения. Возвращает строку «пенсионер» или пустую строку.

2. Функция начисления премий сотрудникам. Входной параметр – базовая ставка. Размер премии вычисляется как произведение базовой ставки на оклад и на коэффициент, который зависит от стажа работы на данном предприятии:

- 0.1, если стаж от 1 до 5 лет;
- 0.2, если стаж от 5 до 10 лет;
- 0.3, если стаж от 10 до 20 лет;
- 0.5, если стаж от 20 до 30 лет;
- 1, если стаж свыше 30 лет.

Результат работы функции добавляется в таблицу «Премии» (поля ФИО сотрудника, стаж, размер премии), которая в начале работы очищается от старых данных. Стаж рассчитывается с учетом трудовой книжки. Людям, проработавшим менее 1 года, премия не выплачивается (и в таблицу они не добавляются). Пенсионерам выплачивается половина премии. Использовать ранее созданную функцию. Создание триггеров.

1. Проверка значений всех полей отношения «Сотрудники», для которых могут быть определены домены (в т.ч., ИНН может содержать только цифры, а дата поступления на работу должна быть не больше текущей даты). Если при вводе данных дата поступления не указана, устанавливать текущую дату.

2. Формирование таблицы «Трудовая книжка»: при изменении занимаемой должности в таблице «Сотрудники», предыдущая должность добавляется в таблицу «Трудовая книжка» с указанием даты начала работы в этой должности (старое значение даты вступления в должность из таблицы «Сотрудники»).

4. Ход работы

Код создания таблиц, генераторов, функций, триггеров и запросов представлен в Приложении А.

sifer	department_name
1	Отдел разработки
2	Отдел маркетинга
3	Отдел кадров
4	Бухгалтерия
(4 rows)	

Рисунок 2 – Таблица отделений

name	salary
Разработчик	500000
Старший разработчик	700000
Менеджер по маркетингу	450000
HR-специалист	400000
Бухгалтер	420000
Директор	1000000
(6 rows)	

Рисунок 3 – Таблица должностей

personnel_number	full_name	iin	gender	birthday
1	Иванов Иван Иванович	123456789012	м	1980-05-
2	Петрова Анна Сергеевна	234567890123	ж	1965-08-
3	Сидоров Петр Васильевич	345678901234	м	1990-12-
4	Козлова Мария Дмитриевна	456789012345	ж	1968-03-
5	Смирнов Алексей Петрович	567890123456	м	1985-11-
6	Федорова Елена Викторовна	678901234567	ж	1958-07-
7	Николаев Дмитрий Олегович	789012345678	м	1995-02-
8	Васильев Геннадий Петрович	890123456789	м	1959-01-
(8 rows)				

Рисунок 4 – Таблица сотрудников

employee	department	position	date_of_receipt
1	1	Разработчик	2015-03-10
2	2	Менеджер по маркетингу	2010-07-15
3	1	Старший разработчик	2018-11-01
4	3	HR-специалист	2005-09-20
5	4	Бухгалтер	2020-01-15
6	1	Директор	2000-06-01
7	2	Менеджер по маркетингу	2022-12-01
(7 rows)			

Рисунок 5 – Таблица трудовых книжек

full_name	experience	bonus_size	accrual_date
Иванов Иван Иванович	15	150000000	2025-11-06
Петрова Анна Сергеевна	30	67500000	2025-11-06
Сидоров Петр Васильевич	10	140000000	2025-11-06
Козлова Мария Дмитриевна	30	100000000	2025-11-06
Смирнов Алексей Петрович	5	84000000	2025-11-06
Федорова Елена Викторовна	45	250000000	2025-11-06
Николаев Дмитрий Олегович	2	45000000	2025-11-06
Васильев Геннадий Петрович	32	50000000	2025-11-06

(8 rows)

Рисунок 6 – Таблица премий

В Листинге 1 представлены генераторы. Их работу можно увидеть на Рис. 2 (генерация поля sifer) и Рис. 4 (генерация поля personnel_number).

Листинг 1 – Генераторы

```
CREATE SEQUENCE IF NOT EXISTS department_sifer_gen
  START WITH 1
  INCREMENT BY 1;

CREATE SEQUENCE IF NOT EXISTS employee_personnel_number_gen
  START WITH 1
  INCREMENT BY 1;
```

В Листинге 2 представлены функции проверки пенсионер ли сотрудник или нет и функции расчета премий.

Листинг 2 – Функции

```
CREATE OR REPLACE FUNCTION is_pensioner(gender char(1), birthday_date date)
  RETURNS VARCHAR(10) AS $$
DECLARE
  pension_age INTEGER;
  current_age INTEGER;
BEGIN
  current_age := EXTRACT(YEAR FROM AGE(CURRENT_DATE, birthday_date));

  IF gender = 'м' THEN
    pension_age := 60;
  ELSE
    pension_age := 55;
  END IF;

  IF current_age >= pension_age THEN
    RETURN 'пенсионер';
  ELSE
    RETURN '';
  END IF;
END;
```

```

        END IF;

END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION calc_bonuses(base_salary INTEGER)
    RETURNS VOID AS $$
DECLARE
    employee RECORD;
    bonus_mult DECIMAL;
    bonus_size INTEGER;
    experience INTEGER;
BEGIN

    DELETE FROM bonuses;

    FOR employee IN
        SELECT
            e.full_name,
            e.gender,
            e.birthday_date,
            e.date_of_receipt,
            e.previous_experience,
            p.salary
        FROM employees AS e
        JOIN positions AS p ON e.position = p.name
    LOOP

        experience := EXTRACT(YEAR FROM age(CURRENT_DATE, employee.date_of_receipt));

        CONTINUE WHEN experience < 1;

        IF experience > 30 THEN
            bonus_mult := 1.0;
        ELSIF experience >= 20 THEN
            bonus_mult := 0.5;
        ELSIF experience >= 10 THEN
            bonus_mult := 0.3;
        ELSIF experience >= 5 THEN
            bonus_mult := 0.2;
        ELSIF experience >= 1 THEN
            bonus_mult := 0.1;
        ELSE
            bonus_mult := 0.0;
        END IF;

        bonus_size := base_salary * bonus_mult * employee.salary;

        IF is_pensioner(employee.gender, employee.birthday_date) = 'пенсионер' THEN
            bonus_size := bonus_size / 2;
        END IF;

        INSERT INTO bonuses (full_name, experience, bonus_size, accrual_date)
            VALUES (employee.full_name, experience + employee.previous_experience, bonus_size,
CURRENT_DATE);

    END LOOP;

END;
$$ LANGUAGE plpgsql;

```

Листинг 3 – Вызов функций

```

SELECT full_name, birthday_date, gender, is_pensioner(gender, birthday_date) as pension_status
FROM employees;

```

```
SELECT calc_bonuses(1000);  
SELECT * FROM bonuses;
```

5. Вывод

ПРИЛОЖЕНИЕ А

Листинг 4 – Файл init.sql

```
\i drop_all.sql

CREATE TABLE departments (
    sifer INTEGER NOT NULL,
    department_name VARCHAR(30) NOT NULL,
    PRIMARY KEY (sifer)
);

CREATE TABLE positions (
    name VARCHAR(30) NOT NULL,
    salary INTEGER NOT NULL,
    CHECK (salary > 0),
    PRIMARY KEY (name)
);

CREATE TABLE employees (
    personnel_number INTEGER NOT NULL,
    full_name VARCHAR(30) NOT NULL,
    IIN CHAR(12) NOT NULL,
    gender CHAR(1) NOT NULL DEFAULT 'ж',
    birthday_date DATE NOT NULL,
    department INTEGER,
    position VARCHAR(30),
    date_of_receipt DATE,
    previous_experience INTEGER NOT NULL,
    CHECK (gender IN ('м', 'ж')),
    PRIMARY KEY (personnel_number),
    FOREIGN KEY (department) REFERENCES departments (sifer)
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    FOREIGN KEY (position) REFERENCES positions (name)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

CREATE TABLE working_book (
    employee INTEGER NOT NULL,
    department INTEGER NOT NULL,
    position VARCHAR(30) NOT NULL,
    date_of_receipt DATE NOT NULL,
    FOREIGN KEY (employee) REFERENCES employees (personnel_number)
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    FOREIGN KEY (department) REFERENCES departments (sifer)
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    FOREIGN KEY (position) REFERENCES positions (name)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

CREATE TABLE bonuses (
    full_name VARCHAR(30) NOT NULL,
    experience INTEGER NOT NULL,
    bonus_size INTEGER NOT NULL,
    accrual_date DATE
);
```

```

\i generators.sql
\i functions.sql
\i triggers.sql

\i data.sql

```

Листинг 5 – Файл triggers.sql

```

-- 1. Проверка значений всех полей отношения «Сотрудники», для которых могут быть
-- определены домены (в т.ч., ИНН может содержать только цифры, а дата поступления на работу
-- должна быть не больше текущей даты). Если при вводе данных дата поступления не указана,
-- устанавливать текущую дату.

CREATE OR REPLACE FUNCTION verify_employee()
    RETURNS TRIGGER AS $$
BEGIN

    IF NEW.IIN !~ '^[0-9]{12}$' THEN
        RAISE NOTICE 'ИНН должен содержать только 12 цифр: %', NEW.IIN;
        RETURN NULL;
    END IF;

    IF EXTRACT(YEAR FROM AGE(CURRENT_DATE, NEW.birthday_date)) < 16 THEN
        RAISE NOTICE 'сотрудник должен быть старше 16 лет: %', NEW.birthday_date;
        RETURN NULL;
    END IF;

    IF NEW.previous_experience < 0 THEN
        RAISE NOTICE 'стаж на предыдущих позициях должен быть положительным: %',
NEW.previous_experience;
        RETURN NULL;
    END IF;

    IF NEW.date_of_receipt > CURRENT_DATE THEN
        RAISE NOTICE 'дата подачи заявления должна быть меньше текущей: %', NEW.date_of_receipt;
        NEW.date_of_receipt := CURRENT_DATE;
    END IF;

    IF NEW.date_of_receipt IS NULL THEN
        RAISE NOTICE 'отсутствует дата подачи заявления - вставляется текущая дата: %',
CURRENT_DATE;
        NEW.date_of_receipt := CURRENT_DATE;
    END IF;

    RETURN NEW;

END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER tr_verify_employee
    BEFORE INSERT ON employees
    FOR EACH ROW
    EXECUTE PROCEDURE verify_employee();

-- 2. Формирование таблицы «Трудовая книжка»: при изменении занимаемой должности в
-- таблице «Сотрудники», предыдущая должность добавляется в таблицу «Трудовая книжка» с
-- указанием даты начала работы в этой должности (старое значение даты вступления в должность
-- из таблицы «Сотрудники»).

CREATE OR REPLACE FUNCTION record_position()
    RETURNS TRIGGER AS $$

```

```

BEGIN

    IF NEW.position = OLD.position THEN
        RAISE NOTICE 'должность должна отличаться: new = %, old = %', NEW.position, OLD.position;
        RETURN NULL;
    END IF;

    INSERT INTO working_book (
        employee,
        department,
        position,
        date_of_receipt
    ) VALUES (
        OLD.personnel_number,
        OLD.department,
        OLD.position,
        OLD.date_of_receipt
    );

    NEW.date_of_receipt := CURRENT_DATE;

    RETURN NEW;

END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER tr_record_position
BEFORE UPDATE ON employees
FOR EACH ROW
EXECUTE PROCEDURE record_position();

```

Листинг 6 – Файл data.sql

```
\i delete_data.sql
```

```

INSERT INTO departments (sifer, department_name) VALUES
(NEXTVAL('department_sifer_gen'), 'Отдел разработки'),
(NEXTVAL('department_sifer_gen'), 'Отдел маркетинга'),
(NEXTVAL('department_sifer_gen'), 'Отдел кадров'),
(NEXTVAL('department_sifer_gen'), 'Бухгалтерия');

INSERT INTO positions (name, salary) VALUES
('Разработчик', 500000),
('Старший разработчик', 700000),
('Менеджер по маркетингу', 450000),
('HR-специалист', 400000),
('Бухгалтер', 420000),
('Директор', 1000000);

INSERT INTO employees (personnel_number, full_name, IIN, gender, birthday_date, department,
position, date_of_receipt, previous_experience) VALUES
(NEXTVAL('employee_personnel_number_gen'), 'Иванов Иван Иванович', '123456789012', 'м',
'1980-05-15', 1, 'Разработчик', '2015-03-10', 5),
(NEXTVAL('employee_personnel_number_gen'), 'Петрова Анна Сергеевна', '234567890123', 'ж',
'1965-08-20', 2, 'Менеджер по маркетингу', '2010-07-15', 15),
(NEXTVAL('employee_personnel_number_gen'), 'Сидоров Петр Васильевич', '345678901234', 'м',
'1990-12-03', 1, 'Старший разработчик', '2018-11-01', 3),
(NEXTVAL('employee_personnel_number_gen'), 'Козлова Мария Дмитриевна', '456789012345', 'ж',
'1968-03-25', 3, 'HR-специалист', '2005-09-20', 10),
(NEXTVAL('employee_personnel_number_gen'), 'Смирнов Алексей Петрович', '567890123456', 'м',
'1985-11-12', 4, 'Бухгалтер', '2020-01-15', 0),
(NEXTVAL('employee_personnel_number_gen'), 'Федорова Елена Викторовна', '678901234567', 'ж',
'1958-07-30', 1, 'Директор', '2000-06-01', 20),

```

```
(NEXTVAL('employee_personnel_number_gen'), 'Николаев Дмитрий Олегович', '789012345678', 'м',  
'1995-02-14', 2, 'Менеджер по маркетингу', '2022-12-01', 0),  
(NEXTVAL('employee_personnel_number_gen'), 'Васильев Геннадий Петрович', '890123456789', 'м',  
'1959-01-10', 1, 'Разработчик', '2018-06-15', 25);
```

```
INSERT INTO working_book (employee, department, position, date_of_receipt) VALUES  
(1, 1, 'Разработчик', '2015-03-10'),  
(2, 2, 'Менеджер по маркетингу', '2010-07-15'),  
(3, 1, 'Старший разработчик', '2018-11-01'),  
(4, 3, 'HR-специалист', '2005-09-20'),  
(5, 4, 'Бухгалтер', '2020-01-15'),  
(6, 1, 'Директор', '2000-06-01'),  
(7, 2, 'Менеджер по маркетингу', '2022-12-01');
```

Листинг 7 – Файл delete_data.sql

```
DELETE FROM bonuses;  
DELETE FROM working_book;  
DELETE FROM employees;  
DELETE FROM positions;  
DELETE FROM departments;  
  
ALTER SEQUENCE department_sifer_gen RESTART WITH 1;  
ALTER SEQUENCE employee_personnel_number_gen RESTART WITH 1;
```

Листинг 8 – Файл drop_all.sql

```
DROP TABLE bonuses;  
DROP TABLE working_book;  
DROP TABLE employees;  
DROP TABLE positions;  
DROP TABLE departments;
```