-Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Институт информационных технологий

Кафедра ИС

# ОТЧЕТ

по лабораторной работе №2

ЯЗЫК SQL. ГЕНЕРАТОРЫ. ФУНКЦИИ. ТРИГГЕРЫ

Выполнил:

ст. гр. ИС/б-21-2-о

Мовенко К. М.

Проверила:

Абрамович А.Ю.

Севастополь

2023

# ЦЕЛЬ РАБОТЫ

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

# ЗАДАНИЕ

В соответствии с вариантом задания (Рисунок 2.1) создать базу данных, соответствующие функции и триггеры, а также генератор для каждого первичного ключа (исключение, если первичный ключ является символьным полем).

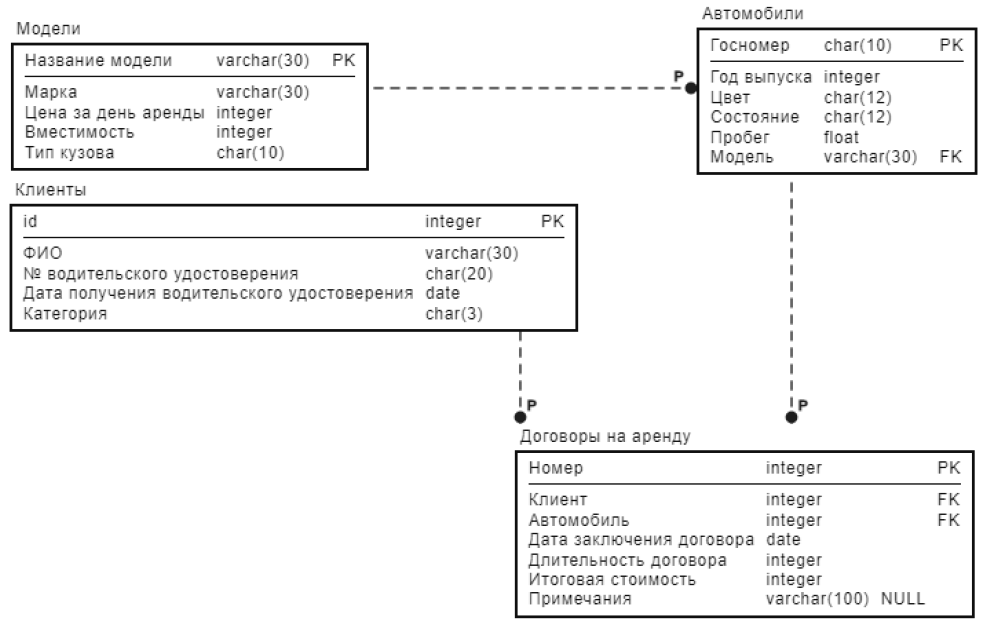


Рисунок 2.1 – Вариант 8, схема базы данных

# Создание функций:

1. Функция, принимающая два параметра – дату и продолжительность аренды. Функция должна возвращать предположительную дату возврата автомобиля.

2. Функция, добавляющая в специальную таблицу «Отчёт» сведения о каждом автомобиле определенной марки на текущую дату. Параметр – марка автомобиля. Функция должна перед началом работы очищать таблицу. Для автомобилей, находящихся в прокате, функция должна вносить номер, год выпуска, цвет, пробег, ФИО арендатора, начало и предполагаемую дату возврата автомобиля. Для автомобилей, не находящихся в прокате, функция должна вносить номер, год выпуска, цвет, пробег, дату последнего возврата из аренды и продолжительность аренды за последний год (в днях). Использовать ранее созданную функцию.

# Создание триггеров:

1. Триггер, устанавливающий стоимость проката после установки даты возврата автомобиля как произведение стоимости аренды на её реальную продолжительность;

2. Проверка значений всех полей отношения «Договоры на аренду», для которых могут быть определены домены: дата начала договора не меньше текущей, дата возврата автомобиля не определена, стоимость договора равна продолжительности, умноженной на стоимость аренды; продолжительность больше 0;

# ХОД РАБОТЫ

В среде pgAdmin 4 была создана база данных на PostgreSQL. В ней была создана таблица models для хранения информации о моделях авто, представленных в автопрокате.

CREATE TABLE IF NOT EXISTS public.models

(

name character varying(30) COLLATE pg\_catalog."default" NOT NULL,

brand character varying(30) COLLATE pg\_catalog."default" NOT NULL,

rent\_price integer NOT NULL,

capacity integer,

body\_type character(10) COLLATE pg\_catalog."default",

CONSTRAINT models\_pkey PRIMARY KEY (name)

)

Таблица models была заполнена рядом значений (Рисунок 3.1).

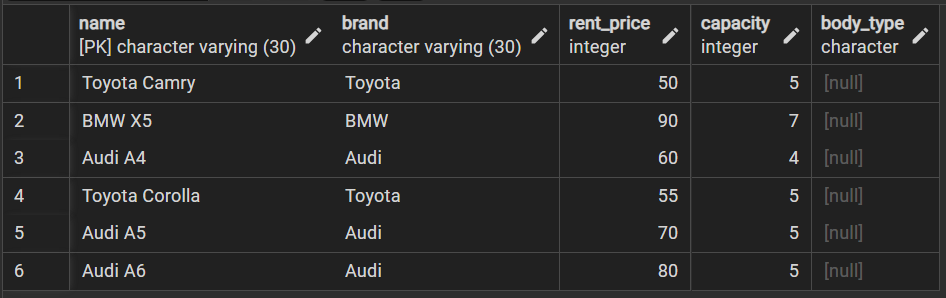


Рисунок 3.1 – Заполнение таблицы models

Была создана таблица autos для хранения информации о конкретных экземплярах выдаваемых в аренду автомобилей. Поле model установлено внешним ключом на поле name таблицы models.

CREATE TABLE IF NOT EXISTS public.autos

(

"number" character(10) COLLATE pg\_catalog."default" NOT NULL,

year integer NOT NULL,

color character(12) COLLATE pg\_catalog."default" NOT NULL,

condition character(12) COLLATE pg\_catalog."default" DEFAULT 'undefined'::bpchar,

mileage real NOT NULL,

model character varying(30) COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT autos\_pkey PRIMARY KEY ("number"),

CONSTRAINT fk\_auto\_model FOREIGN KEY (model)

REFERENCES public.models (name) MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE CASCADE

NOT VALID

)

Таблица autos и была заполнена рядом значений (Рисунок 3.2).

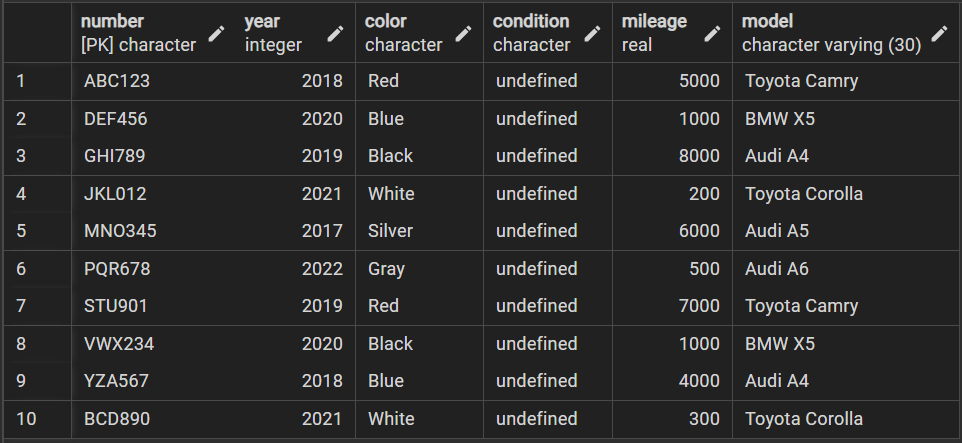


Рисунок 3.2 – Заполнение таблицы autos

Была создана таблица clients для хранения информации о клиентах автопроката.

CREATE TABLE IF NOT EXISTS public.clients

(

id integer NOT NULL DEFAULT nextval('clients\_id'::regclass),

full\_name character varying(30) COLLATE pg\_catalog."default" NOT NULL,

license character(20) COLLATE pg\_catalog."default",

cert\_date date,

category character(3) COLLATE pg\_catalog."default",

CONSTRAINT clients\_pkey PRIMARY KEY (id)

)

Поле id таблицы clients по умолчанию заполняется генератором clients\_id.

CREATE SEQUENCE IF NOT EXISTS public.clients\_id

INCREMENT 1

START 1

MINVALUE 1

MAXVALUE 9223372036854775807

CACHE 1

OWNED BY clients.id;

Таблица clients была заполнена рядом значений (Рисунок 3.3).

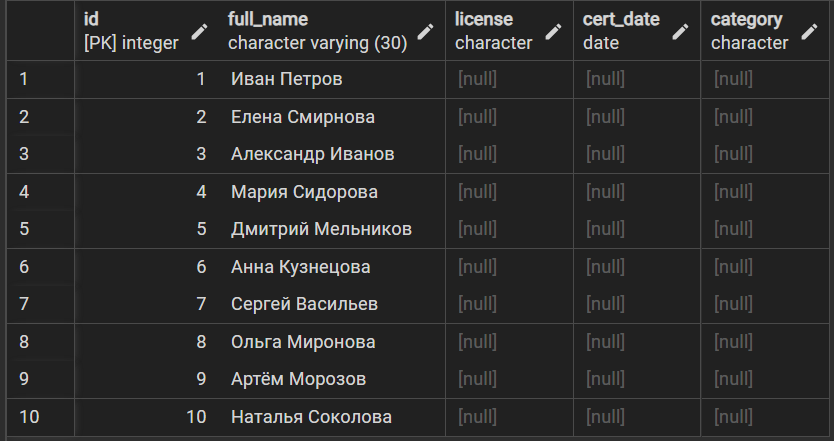


Рисунок 3.3 – Заполнение таблицы clients

Была создана таблица rent\_contracts для хранения информации о заказах на аренду автомобилей и их состояниях. Поле auto является внешним ключом на таблицу autos, а поле client – на таблицу clients.

CREATE TABLE IF NOT EXISTS public.rent\_contracts

(

id integer NOT NULL DEFAULT nextval('contract\_id'::regclass),

client integer NOT NULL,

auto character(10) COLLATE pg\_catalog."default" NOT NULL,

start\_date date NOT NULL,

duration integer NOT NULL,

end\_date date,

total\_cost integer NOT NULL DEFAULT 0,

notes character varying(100) COLLATE pg\_catalog."default" DEFAULT NULL::character varying,

CONSTRAINT rent\_contracts\_pkey PRIMARY KEY (id),

CONSTRAINT fk\_contract\_auto FOREIGN KEY (auto)

REFERENCES public.autos ("number") MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE CASCADE

NOT VALID,

CONSTRAINT fk\_contract\_client FOREIGN KEY (client)

REFERENCES public.clients (id) MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE CASCADE

NOT VALID

)

Поле id таблицы rent\_contracts по умолчанию заполняется генератором contract\_id.

CREATE SEQUENCE IF NOT EXISTS public.contract\_id

INCREMENT 1

START 1

MINVALUE 1

MAXVALUE 9223372036854775807

CACHE 1

OWNED BY rent\_contracts.id;

Для таблицы rent\_contracts был установлен триггер contracts\_insert\_new. При добавлении нового контракта он вызывает триггерную функцию check\_new\_contract.

CREATE OR REPLACE TRIGGER contracts\_insert\_new

BEFORE INSERT

ON public.rent\_contracts

FOR EACH ROW

EXECUTE FUNCTION public.check\_new\_contract();

Функция check\_new\_contract проверяет, чтобы дата заключения договора не была меньше текущей, чтобы дата возврата по умолчанию равнялась NULL и чтобы продолжительность была больше нуля.

CREATE OR REPLACE FUNCTION public.check\_new\_contract()

RETURNS trigger

LANGUAGE 'plpgsql'

COST 100

VOLATILE NOT LEAKPROOF

AS $BODY$

BEGIN

-- Проверка даты начала договора

IF NEW.start\_date < CURRENT\_DATE THEN

RAISE EXCEPTION 'Дата начала договора должна быть не меньше текущей';

END IF;

-- Проверка отсутствия даты возврата автомобиля

IF NEW.end\_date IS NOT NULL THEN

RAISE EXCEPTION 'Дата возврата не может быть определена изначально';

END IF;

-- Проверка продолжительности

IF NEW.duration <= 0 THEN

RAISE EXCEPTION 'Продолжительность договора должна быть больше нуля';

END IF;

-- Установка стоимости договора

NEW.total\_cost := NEW.duration \*

(SELECT models.rent\_price FROM models

JOIN autos ON models.name = autos.model

WHERE autos.number = NEW.auto);

RETURN NEW;

END;

$BODY$;

Также для таблицы rent\_contracts был установлен триггер contracts\_update\_end\_year, который при обновлении атрибута end\_year (который изначально равен NULL) вызывает триггерную функцию update\_total\_cost.

CREATE OR REPLACE TRIGGER contracts\_update\_end\_year

BEFORE UPDATE OF end\_date

ON public.rent\_contracts

FOR EACH ROW

EXECUTE FUNCTION public.update\_total\_cost();

Функция update\_total\_cost проверяет, чтобы дата возврата не была меньше даты заключения контракта, а также обновляет итоговую стоимость аренды относительно её реальной продолжительности (разницы между датами начала аренды и возврата авто).

CREATE OR REPLACE FUNCTION public.update\_total\_cost()

RETURNS trigger

LANGUAGE 'plpgsql'

COST 100

VOLATILE NOT LEAKPROOF

AS $BODY$

DECLARE

price INTEGER;

duration INTEGER;

BEGIN

-- Проверка даты возврата

IF NEW.end\_date < NEW.start\_date THEN

RAISE EXCEPTION 'Дата возврата меньше даты заключения контракта';

END IF;

-- Определение стоимости аренды

SELECT rent\_price INTO price

FROM models

JOIN autos ON models.name = autos.model

WHERE autos.number = NEW.auto;

-- Определение стоимости по "реальной" продолжительности

duration := NEW.end\_date - NEW.start\_date;

NEW.total\_cost := price \* (duration);

RETURN NEW;

END;

$BODY$;

Таблица rent\_contracts была заполнена рядом значений на основе данных предыдущих таблиц (3.4).

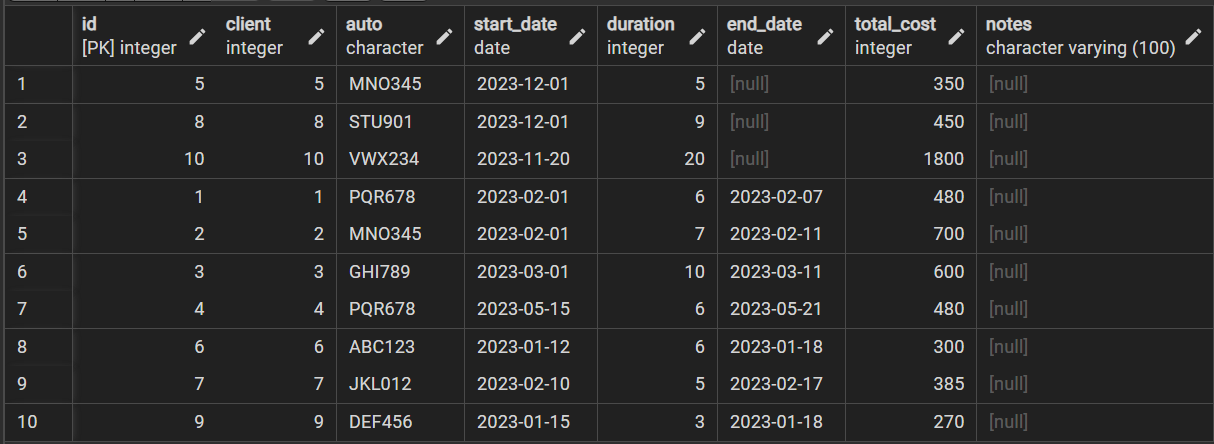


Рисунок 3.4 – Заполнение таблицы rent\_contracts

Были протестированы триггеры, обеспечивающие целостность данных при добавлении (часть ограничений изначально была снята). Так при добавлении новых контрактов проверяется, что дата возврата не определена (Рисунок 3.5), дата заключение договора не меньше текущей (Рисунок 3.6), договор был завершён не раньше, чем был заключён (Рисунок 3.7).

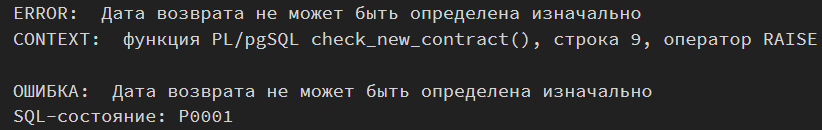


Рисунок 3.5 – Проверка неопределённости даты возврата при добавлении нового контракта

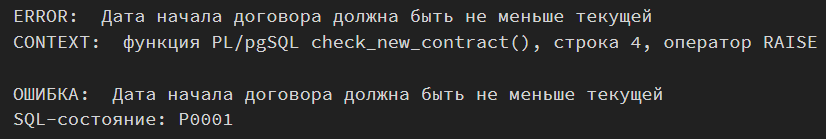


Рисунок 3.6 – Проверка даты начала договора

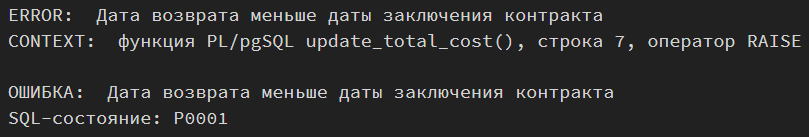


Рисунок 3.7 – Проверка даты окончания договора

Была создана таблица report, через которую в дальнейшем будут отображаться отчёты об автомобилях заданной марки.

CREATE TABLE IF NOT EXISTS public.report

(

"number" character(10) COLLATE pg\_catalog."default" NOT NULL,

year integer NOT NULL,

color character(12) COLLATE pg\_catalog."default" NOT NULL,

mileage real NOT NULL,

tenant character varying(30) COLLATE pg\_catalog."default",

start\_date date,

end\_date date,

duration integer,

CONSTRAINT report\_pkey PRIMARY KEY ("number")

)

Была написана функция get\_expiry\_date, которая складывает начальную дату с продолжительность, заданной в днях, и возвращает конечную дату.

CREATE OR REPLACE FUNCTION public.get\_expiry\_date(

start\_date date,

duration integer)

RETURNS date

LANGUAGE 'plpgsql'

COST 100

VOLATILE PARALLEL UNSAFE

AS $BODY$

DECLARE

BEGIN

RETURN start\_date + duration;

END;

$BODY$

Была написана функция make\_report, которая создаёт в таблице report отчёт о состоянии автомобилей заданной марки.

CREATE OR REPLACE FUNCTION public.make\_report(

brand\_name character varying)

RETURNS void

LANGUAGE 'plpgsql'

COST 100

VOLATILE PARALLEL UNSAFE

AS $BODY$

DECLARE

BEGIN

-- Очистка таблицы report

DELETE FROM public.report;

-- Информация об автомобилях, находящихся в аренде

INSERT INTO public.report(number, year, color, mileage, tenant, start\_date, end\_date)

SELECT

a.number,

a.year,

a.color,

a.mileage,

cl.full\_name,

rc.start\_date,

get\_expiry\_date(rc.start\_date, rc.duration)

FROM

public.autos AS a

JOIN public.rent\_contracts AS rc ON a.number = rc.auto

JOIN public.clients AS cl ON rc.client = cl.id

JOIN public.models AS m ON m.name = a.model

WHERE

m.brand = brand\_name

AND rc.end\_date IS NULL;

-- Информация об автомобилях, никогда не бывших в аренде

INSERT INTO public.report (number, year, color, mileage)

SELECT

a.number,

a.year,

a.color,

a.mileage

FROM

public.autos AS a

JOIN public.models AS m ON m.name = a.model

WHERE

m.brand = brand\_name

AND a.number NOT IN (SELECT auto FROM public.rent\_contracts);

-- Информация об автомобилях, бывших в аренде ранее

INSERT INTO public.report (number, year, color, mileage, end\_date, duration)

SELECT

a.number,

a.year,

a.color,

a.mileage,

MAX(rc.end\_date),

SUM(rc.duration)

FROM

public.autos AS a

JOIN public.rent\_contracts AS rc ON a.number = rc.auto

JOIN public.models AS m ON m.name = a.model

WHERE

m.brand = brand\_name

AND rc.end\_date >= CURRENT\_DATE - INTERVAL '1 year'

AND NOT EXISTS (

SELECT \*

FROM public.rent\_contracts AS rc2

WHERE rc2.auto = a.number

AND rc2.end\_date IS NULL

)

GROUP BY

a.number;

END;

$BODY$;

Был составлен отчёт для марки Audi (Рисунок 3.8). В нём можно увидеть три случая: MNO345 (Audi A5) находится в аренде, GHI789 и PQR678 (A4 и A6) находились в аренде ранее (была подсчитана длительность аренды за год), YZA567 (второй A4) ни разу не был в аренде.

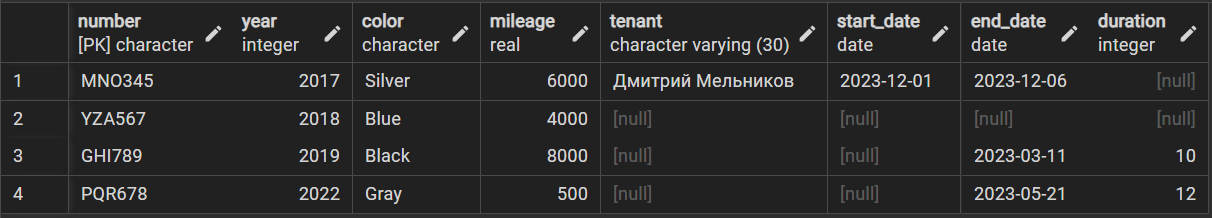


Рисунок 3.8 – Отчёт об автомобилях марки Audi

# ВЫВОД

В ходе работы были изучены следующие компоненты баз данных на SQL: генераторы, функции, триггеры. Они позволяют автоматизировать часть процесса заполнения данных (автоматически генерируемый id, подсчёт значений через функции), а также поддерживать их целостность путём автоматических проверок в ответ на события в БД (триггеры).