МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

ИНСТИТУТ НЕПРЕРЫВНОГО И ДИСТАНЦИОННОГО ОБРАЗОВАНИЯ

КАФЕДРА 43

ОЦЕНКА

ПРЕПОДАВАТЕЛЬ

Старший преподаватель		Н. В. Путилова
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ

«Объектно-реляционные базы данных.
Проектирование и создание»
по дисциплине: проектирование баз данных

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №	Z9431	23.04.2023	П. В. Крылов
	номер группы	подпись, дата	инициалы, фамилия

Студенческий билет № _____

Санкт-Петербург 2023

Задание

Спроектировать физическую модель базы данных, находящуюся в третьей нормальной форме и включающей наследование и хотя бы один пользовательский тип в соответствии с заданным вариантом. Написать соответствующий скрипт создания базы данных.

Вариант 11

магазин обоев и покрытий для стен: типы покрытия (обои, живой камень, краска),ширина рулона, название, коллекция, производитель, ширина, материал, цена, текстура материала, партия, дата поставки партии

а. коллекция обоев, начинающаяся со слова Элегия

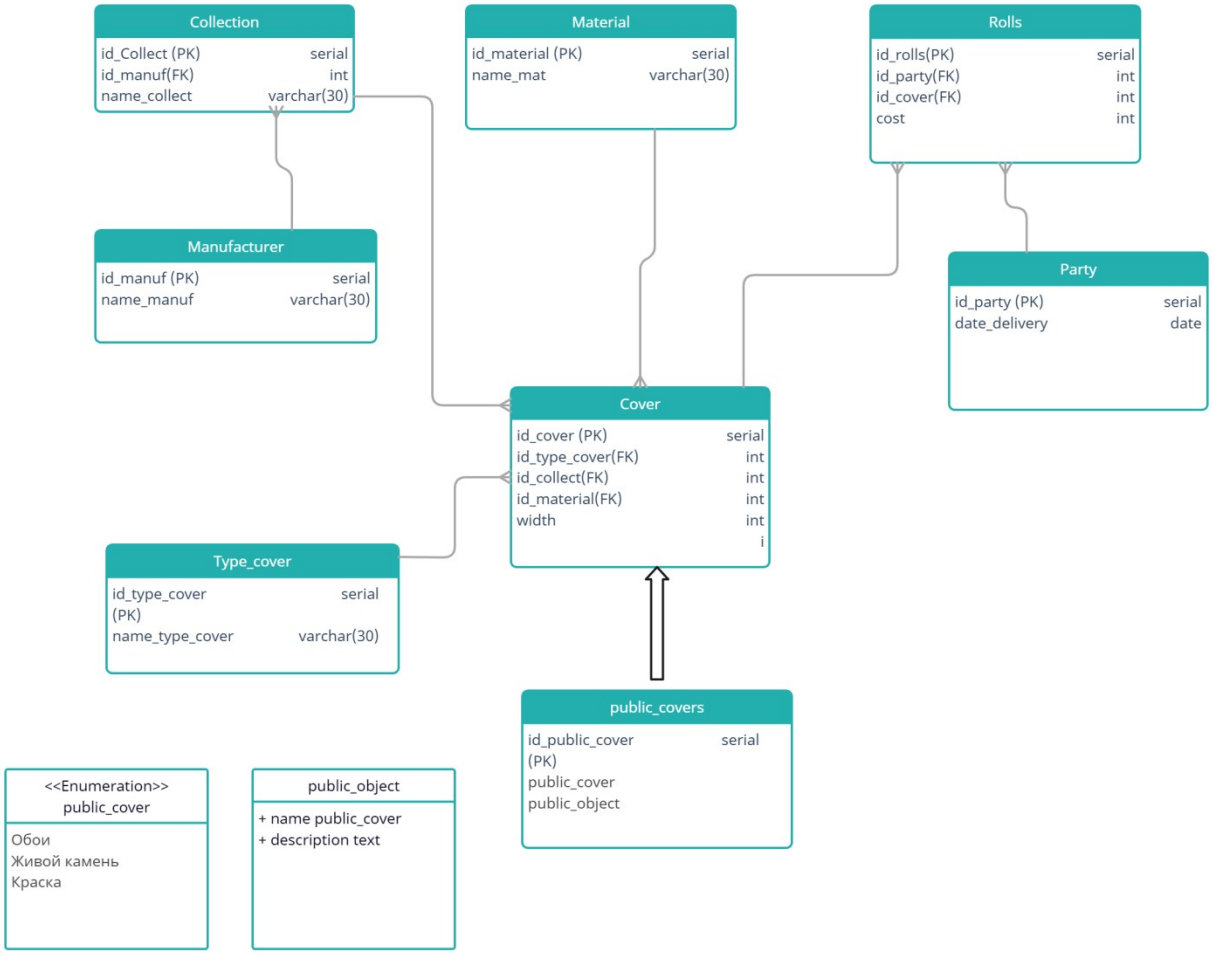
б. производитель, который производит как бумажные, так и флизелиновые обои

в. производитель с самой большой шириной рулона

г. производитель обоев с максимальным количеством коллекций

д. производитель, у которого нет обоев дороже 3000 рублей

Физическая модель базы данных:



Скрипт создания базы данных

```
CREATE DATABASE shop_wallpap
    WITH
    OWNER = postgres
    ENCODING = 'UTF8'
    LC_COLLATE = 'Russian_Russia.1251'
    LC_CTYPE = 'Russian_Russia.1251'
    TABLESPACE = pg_default
    CONNECTION LIMIT = -1;

create type public_cover as enum('обои','живой камень','краска');
create type public_object as (name public_cover,description text);
```

```
create table if not exists public.Manufacturer
(
        id_manuf serial not null primary key,
        name_manuf varchar(30) not null
);
```

```
create table if not exists public.Type_cover
(
	id_type_cover serial not null primary key,
	name_type_cover varchar(30) not null
);


create table if not exists public.Material
(
	id_material serial not null primary key,
	name_mat varchar(30) not null
);


create table if not exists public.Party
(
	id_party serial not null primary key,
	date_delivery date
);


create table if not exists public.Collection
(
	id_collect serial not null primary key,
	name_collect varchar(30) not null,
	id_manuf integer,
	foreign key (id_manuf) references public.manufacturer(id_manuf) on
	delete restrict on update cascade
);


create table if not exists public.Cover
(
	id_cover serial not null primary key,
	id_type_cover integer,
	id_collect integer,
	id_material integer,
	width integer,
	foreign key(id_type_cover) references public."type_cover"(id_type_cover)
on
	delete restrict on update cascade,
	foreign key(id_collect) references public."collection"(id_collect) on
	delete restrict on update cascade,
	foreign key(id_material) references public."material"(id_material) on
```

```
	delete restrict on update cascade
)


create table if not exists public.Rolls
(
	id_rolls serial not null primary key,
	id_party integer,
	id_cover integer,
	cost integer,
	foreign key(id_party) references public."party"(id_party) on
	delete restrict on update cascade,

)


create table if not exists public.public_covers
(
	public_cover public_object not null,
	foreign key(id_collect) references collection(id_collect) on
	delete restrict on update cascade,
	foreign key(id_type_cover) references type_cover(id_type_cover) on
	delete restrict on update cascade,
	foreign key(id_material) references material(id_material) on
	delete restrict on update cascade
)inherits(cover);
```

триггер и функция для ссылочной целостности rolls при удалении или обновлении id_cover

```
CREATE OR REPLACE FUNCTION id_cover_trigger_proc() RETURNS
TRIGGER AS $$
BEGIN
	IF (TG_OP = 'DELETE') THEN
	IF (SELECT count(id_cover) FROM rolls WHERE rolls.id_cover =
old.id_cover) > 0 THEN
			RAISE EXCEPTION 'Запрещено удаление покрытия с этим
id';
		ELSE
			return old;
		END IF;
	ELSEIF (TG_OP = 'UPDATE') THEN
		IF (SELECT count(id_cover) FROM rolls WHERE rolls.id_cover =
old.id_cover) > 0 THEN
			UPDATE rolls SET id_cover = NEW.id_cover WHERE
```

```
id_cover=old.id_cover;
                        return NEW;
                ELSE
                        return NEW;
                END IF;
        END IF;
END
$$ LANGUAGE 'plpgsql';

CREATE TRIGGER id_cover_trigger
BEFORE DELETE OR UPDATE ON cover
FOR EACH ROW EXECUTE PROCEDURE id_cover_trigger_proc();
```