ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Факультет инфокоммуникационных технологий

Дисциплина:

«Проектирование и реализация баз данных»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ No1 «СОЗДАНИЕ БД POSTGRESQL В PGADMIN. РЕЗЕРВНОЕ КОПИРОВАНИЕ И ВОССТАНОВЛЕНИЕ БД»

Выполнил:

студент группы K32402

Пономарев Константин Витальевич

_____

(подпись)

Проверил(а):

Говорова Марина Михайловна

(отметка о выполнении)

_____

(подпись)

Санкт-Петербург 2023 г.

Цель работы 1.1: овладеть практическими навыками установки СУБД PostgreSQL и создания базы данных в pgadmin 4.

Практическое задание 1.1:

1. Установить СУБД PostgreSQL 1X.
2. Создать базу данных с использованием pgadmin 4.

Цель работы 1.2: овладеть практическими навыками создания таблиц базы данных PostgreSQL 1X, заполнения их рабочими данными, резервного копирования и восстановления БД.

Практическое задание 1.2:

1. Создать базу данных с использованием pgAdmin 4 (согласно индивидуальному заданию).
2. Создать схему в составе базы данных.
3. Создать таблицы базы данных.
4. Установить ограничения на данные: Primary Key, Unique, Check, Foreign

   Key.

5. Заполнить таблицы БД рабочими данными.
6. Создать резервную копию БД.

Указание:

Создать две резервные копии:

− с расширением CUSTOM для восстановления БД;

− с расширением PLAIN для листинга (в отчете);

− при создании резервных копий БД настроить параметры Dump options для Type of objects и Queries .

7. Восстановить БД.

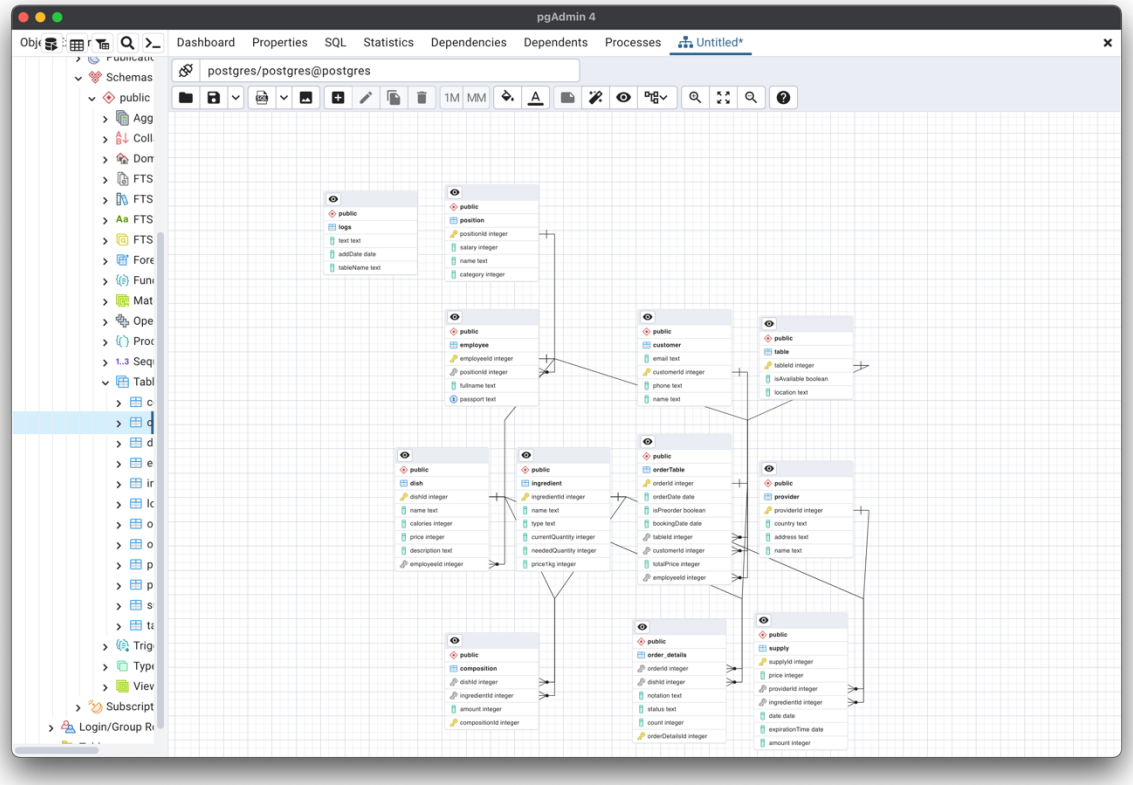

Рисунок 1 – ERD-диаграмма

Dump DB: Сгенерирован в pgAdmin4 и приложен к лабораторной работе в виде файла. Название: postgres_bd_backup

Скрипты работы с базой данных для создания таблиц:

```
-- Table: public.composition

-- DROP TABLE IF EXISTS public.composition;

CREATE TABLE IF NOT EXISTS public.composition
(
    "dishId" integer NOT NULL,
    "ingredientId" integer NOT NULL,
    amount integer NOT NULL,
    "compositionId" integer NOT NULL,
    CONSTRAINT composition_pkey PRIMARY KEY ("compositionId"),
    CONSTRAINT "composition_dishId_fkey" FOREIGN KEY ("dishId")
        REFERENCES public.dish ("dishId") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT "composition_ingredientId_fkey" FOREIGN KEY
("ingredientId")
        REFERENCES public.ingredient ("ingredientId") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.composition
    OWNER to postgres;

-- Table: public.customer

-- DROP TABLE IF EXISTS public.customer;

CREATE TABLE IF NOT EXISTS public.customer
(
    email text COLLATE pg_catalog."default",
    "customerId" integer NOT NULL,
    phone text COLLATE pg_catalog."default" NOT NULL,
    name text COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT guest_pkey PRIMARY KEY ("customerId"),
    CONSTRAINT "check_customerId" CHECK ("customerId" >= 0) NOT
VALID
)
```

```
TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.customer
    OWNER to postgres;

-- Table: public.dish

-- DROP TABLE IF EXISTS public.dish;

CREATE TABLE IF NOT EXISTS public.dish
(
    "dishId" integer NOT NULL,
    name text COLLATE pg_catalog."default" NOT NULL,
    calories integer NOT NULL,
    price integer NOT NULL,
    description text COLLATE pg_catalog."default",
    "employeeId" integer NOT NULL,
    CONSTRAINT dish_pkey PRIMARY KEY ("dishId"),
    CONSTRAINT "dish_employeeId_fkey" FOREIGN KEY ("employeeId")
        REFERENCES public.employee ("employeeId") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.dish
    OWNER to postgres;

-- Table: public.employee

-- DROP TABLE IF EXISTS public.employee;

CREATE TABLE IF NOT EXISTS public.employee
(
    "employeeId" integer NOT NULL,
    "positionId" integer NOT NULL,
    fullname text COLLATE pg_catalog."default" NOT NULL,
    passport text COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT employee_pkey PRIMARY KEY ("employeeId"),
    CONSTRAINT employee_passport_key UNIQUE (passport),
```

```
    CONSTRAINT "employee_positionId_fkey" FOREIGN KEY ("positionId")
        REFERENCES public."position" ("positionId") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.employee
    OWNER to postgres;

-- Table: public.ingredient

-- DROP TABLE IF EXISTS public.ingredient;

CREATE TABLE IF NOT EXISTS public.ingredient
(
    "ingredientId" integer NOT NULL,
    name text COLLATE pg_catalog."default" NOT NULL,
    type text COLLATE pg_catalog."default" NOT NULL,
    "currentQuantity" integer NOT NULL,
    "neededQuantity" integer NOT NULL,
    price1kg integer NOT NULL,
    CONSTRAINT ingredient_pkey PRIMARY KEY ("ingredientId")
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.ingredient
    OWNER to postgres;

-- Table: public.orderTable

-- DROP TABLE IF EXISTS public."orderTable";

CREATE TABLE IF NOT EXISTS public."orderTable"
(
    "orderId" integer NOT NULL,
    "orderDate" date NOT NULL,
    "isPreorder" boolean NOT NULL,
    "bookingDate" date NOT NULL,
    "tableId" integer NOT NULL,
    "customerId" integer NOT NULL,
```

```
    "totalPrice" integer NOT NULL,
    "employeeId" integer NOT NULL,
    CONSTRAINT order_pkey PRIMARY KEY ("orderId"),
    CONSTRAINT "orderTable_employeeId_fkey" FOREIGN KEY
("employeeId")
        REFERENCES public.employee ("employeeId") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT "order_customerId_fkey" FOREIGN KEY ("customerId")
        REFERENCES public.customer ("customerId") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT "order_tableId_fkey" FOREIGN KEY ("tableId")
        REFERENCES public."table" ("tableId") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public."orderTable"
    OWNER to postgres;

-- Table: public.order_details

-- DROP TABLE IF EXISTS public.order_details;

CREATE TABLE IF NOT EXISTS public.order_details
(
    "orderId" integer NOT NULL,
    "dishId" integer NOT NULL,
    notation text COLLATE pg_catalog."default",
    status text COLLATE pg_catalog."default" NOT NULL,
    count integer NOT NULL,
    "orderDetailsId" integer NOT NULL,
    CONSTRAINT order_details_pkey PRIMARY KEY ("orderDetailsId"),
    CONSTRAINT "table_details_dishId_fkey" FOREIGN KEY ("dishId")
        REFERENCES public.dish ("dishId") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT "table_details_orderId_fkey" FOREIGN KEY ("orderId")
        REFERENCES public."orderTable" ("orderId") MATCH SIMPLE
```

```
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.order_details
    OWNER to postgres;

-- Table: public.position

-- DROP TABLE IF EXISTS public."position";

CREATE TABLE IF NOT EXISTS public."position"
(
    "positionId" integer NOT NULL,
    salary integer NOT NULL,
    name text COLLATE pg_catalog."default" NOT NULL,
    category integer NOT NULL,
    CONSTRAINT position_pkey PRIMARY KEY ("positionId"),
    CONSTRAINT check_salary CHECK (salary >= 0) NOT VALID
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public."position"
    OWNER to postgres;

-- Table: public.provider

-- DROP TABLE IF EXISTS public.provider;

CREATE TABLE IF NOT EXISTS public.provider
(
    "providerId" integer NOT NULL,
    country text COLLATE pg_catalog."default" NOT NULL,
    address text COLLATE pg_catalog."default" NOT NULL,
    name text COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT provider_pkey PRIMARY KEY ("providerId")
)

TABLESPACE pg_default;
```

```
ALTER TABLE IF EXISTS public.provider
    OWNER to postgres;

-- Table: public.supply

-- DROP TABLE IF EXISTS public.supply;

CREATE TABLE IF NOT EXISTS public.supply
(
    "supplyId" integer NOT NULL,
    price integer NOT NULL,
    "providerId" integer NOT NULL,
    "ingredientId" integer NOT NULL,
    date date NOT NULL,
    "expirationTime" date NOT NULL,
    amount integer NOT NULL,
    CONSTRAINT supply_pkey PRIMARY KEY ("supplyId"),
    CONSTRAINT "supply_ingredientId_fkey" FOREIGN KEY ("ingredientId")
        REFERENCES public.ingredient ("ingredientId") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT "supply_providerId_fkey" FOREIGN KEY ("providerId")
        REFERENCES public.provider ("providerId") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.supply
    OWNER to postgres;

-- Table: public.table

-- DROP TABLE IF EXISTS public."table";

CREATE TABLE IF NOT EXISTS public."table"
(
    "tableId" integer NOT NULL,
    "isAvailable" boolean NOT NULL,
    location text COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT table_pkey PRIMARY KEY ("tableId")
```

```
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public."table"
    OWNER to postgres;
```

Выводы

В процессе выполнения лабораторной работы удалось более детально познакомиться с работой в pgAdmin 4, получить практические навыки создания таблиц, установки ограничений на таблицы, создания и восстановления резервных копий баз данных.