Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
Факультет инфокоммуникационных технологий

ОТЧЕТ
О ЛАБОРАТОРНОЙ РАБОТЕ № 3

по теме: Создание таблиц базы данных postgresql.
Заполнение таблицрабочими данными.
по дисциплине: Проектирование и реализация баз данных

Специальность:

09.03.03 Мобильные и сетевые технологии

Проверил:

Говорова М.М. _____

Дата: «04» мая 2021г.

Оценка _____

Выполнил:

студент
группы К3241

Коровин А. И.

Санкт-Петербург 2021 г.

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

Оборудование: компьютерный класс.

Программное обеспечение: СУБД PostgreSQL 1X, pgAdmin 4.

Практическое задание:

1. Создать базу данных с использованием pgAdmin 4 (согласноиндивидуальному заданию).
2. Создать схему в составе базы данных.
3. Создать таблицы базы данных.
4. Установить ограничения на данные: *Primary Key, Unique, Check, ForeignKey*.
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 optionsдля Type of objects и Queries* .

7. Восстановить БД.

Индивидуальное задание:

Вариант №13 “Ресторан”

ВЫПОЛНЕНИЕ

1. Название БД

«Restaurant1»

2. Схема инфологической модели данных БД

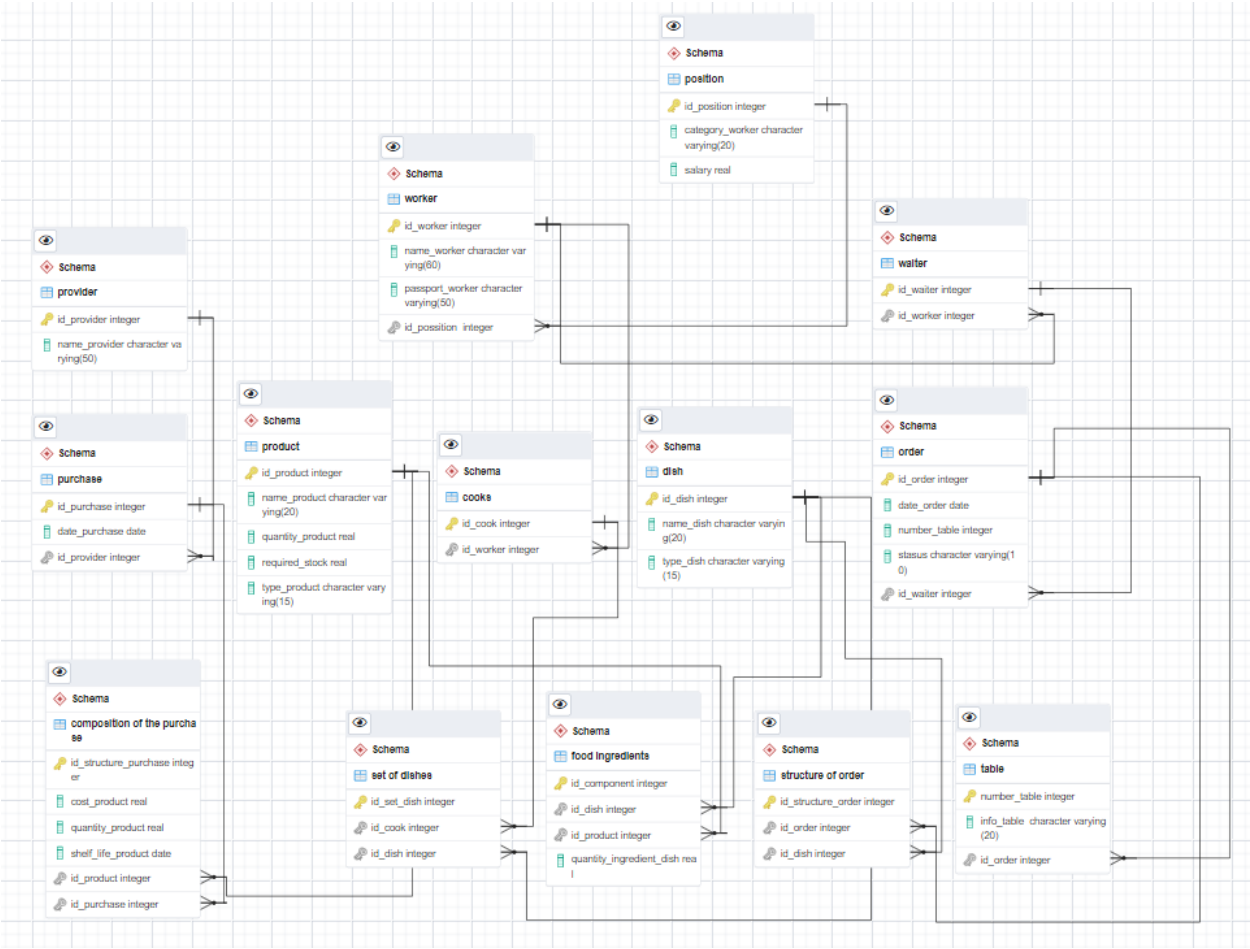

Рисунок 1 – Схема инфологической модели БД, сгенерированная в Generate ERD

3. Plain dump

```
-- Создание базы данных
CREATE DATABASE "Restaurant1"
    WITH
    OWNER = postgres
    ENCODING = 'UTF8'
    LC_COLLATE = 'Russian_Russia.1251'
    LC_CTYPE = 'Russian_Russia.1251'
```

```
    TABLESPACE = pg_default
    CONNECTION LIMIT = -1;


-- Создание схемы
CREATE SCHEMA "Schema"
    AUTHORIZATION postgres;


-- Создание таблицы composition of the purchase и определение ограничений
CREATE TABLE "Schema"."composition of the purchase"
(
    id_structure_purchase integer NOT NULL,
    cost_product real NOT NULL,
    quantity_product real NOT NULL,
    shelf_life_product date NOT NULL,
    id_product integer NOT NULL,
    id_purchase integer NOT NULL,
    CONSTRAINT "composition of the purchase_pkey" PRIMARY KEY (id_structure_purchase),
    CONSTRAINT "composition of the purchase_id_product_fkey" FOREIGN KEY (id_product)
        REFERENCES "Schema".product (id_product) MATCH SIMPLE
        ON UPDATE RESTRICT
        ON DELETE RESTRICT,
    CONSTRAINT "composition of the purchase_id_purchase_fkey" FOREIGN KEY (id_purchase)
        REFERENCES "Schema".purchase (id_purchase) MATCH SIMPLE
        ON UPDATE RESTRICT
        ON DELETE RESTRICT,
    CONSTRAINT "composition of the purchase_quantity_product_check" CHECK (quantity_product >= 0::double precision),
```

```
    CONSTRAINT "composition of the purchase_id_structure_purchase_check" CHECK (id_structure_purchase >= 0)
)
ALTER TABLE "Schema"."composition of the purchase"
    OWNER to postgres;

-- Создание таблицы cooks и определение ограничений
CREATE TABLE "Schema".cooks
(
    id_cook integer NOT NULL,
    id_worker integer NOT NULL,
    CONSTRAINT cooks_pkey PRIMARY KEY (id_cook),
    CONSTRAINT cooks_id_worker_fkey FOREIGN KEY (id_worker)
        REFERENCES "Schema".worker (id_worker) MATCH SIMPLE
        ON UPDATE RESTRICT
        ON DELETE RESTRICT,
    CONSTRAINT cooks_id_cook_check CHECK (id_cook >= 0)
)
ALTER TABLE "Schema".cooks
    OWNER to postgres;

-- Создание таблицы dish и определение ограничений
CREATE TABLE "Schema".dish
(
    id_dish integer NOT NULL,
    name_dish character varying(20) COLLATE pg_catalog."default" NOT NULL,
    type_dish character varying(15) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT dish_pkey PRIMARY KEY (id_dish),
```

```
    CONSTRAINT dish_id_dish_check CHECK (id_dish >= 0),
    CONSTRAINT dish_name_dish_check CHECK (name_dish::text = ANY
(ARRAY['карбонара'::character varying, 'стейк'::character varying,
'овсянка'::character varying, 'борщ'::character varying, 'яичница'::character
varying, 'котлеты'::character varying, 'цыпленок'::character varying, 'тар-
тар'::character varying, 'пицца'::character varying]::text[]))
)
ALTER TABLE "Schema".dish
    OWNER to postgres;

-- Создание таблицы food ingredients и определение ограничений
CREATE TABLE "Schema"."food ingredients"
(
    id_component integer NOT NULL,
    id_dish integer NOT NULL,
    id_product integer NOT NULL,
    quantity_ingredient_dish real NOT NULL,
    CONSTRAINT "food ingredients_pkey" PRIMARY KEY (id_component),
    CONSTRAINT "food ingredients_id_dish_fkey" FOREIGN KEY (id_dish)
        REFERENCES "Schema".dish (id_dish) MATCH SIMPLE
        ON UPDATE RESTRICT
        ON DELETE RESTRICT,
    CONSTRAINT "food ingredients_id_product_fkey" FOREIGN KEY
(id_product)
        REFERENCES "Schema".product (id_product) MATCH SIMPLE
        ON UPDATE RESTRICT
        ON DELETE RESTRICT,
    CONSTRAINT "food ingredients_id_component_check" CHECK
(id_component >= 0),
    CONSTRAINT "food ingredients_quantity_ingredient_dish_check" CHECK
(quantity_ingredient_dish >= 0::double precision)
```

```
)
ALTER TABLE "Schema"."food ingredients"
    OWNER to postgres;

-- Создание таблицы order и определение ограничений
CREATE TABLE "Schema"."order"
(
    id_order integer NOT NULL,
    date_order date NOT NULL,
    number_table integer NOT NULL,
    stasus character varying(10) COLLATE pg_catalog."default" NOT NULL,
    id_waiter integer NOT NULL,
    CONSTRAINT order_pkey PRIMARY KEY (id_order),
    CONSTRAINT order_id_waiter_fkey FOREIGN KEY (id_waiter)
        REFERENCES "Schema".waiter (id_waiter) MATCH SIMPLE
        ON UPDATE RESTRICT
        ON DELETE RESTRICT,
    CONSTRAINT order_id_order_check CHECK (id_order >= 0),
    CONSTRAINT order_number_table_check CHECK (number_table >= 0
AND number_table <= 15)
)
ALTER TABLE "Schema"."order"
    OWNER to postgres;

-- Создание таблицы position и определение ограничений
CREATE TABLE "Schema"."position"
(
    id_position integer NOT NULL,
    category_worker character varying(20) COLLATE pg_catalog."default" NOT
```

```
NULL,
    salary real NOT NULL,
    CONSTRAINT position_pkey PRIMARY KEY (id_position),
    CONSTRAINT position_id_position_check CHECK (id_position >= 0),
    CONSTRAINT position_category_worker_check CHECK (category_worker::text = ANY (ARRAY['администратор'::character varying, 'повар'::character varying, 'директор'::character varying, 'уборщица'::character varying, 'официант'::character varying, 'бухгалтер'::character varying]::text[])),
    CONSTRAINT position_id_position_check1 CHECK (id_position >= 0)
)
ALTER TABLE "Schema"."position"
    OWNER to postgres;


-- Создание таблицы product и определение ограничений
CREATE TABLE "Schema".product
(
    id_product integer NOT NULL,
    name_product character varying(20) COLLATE pg_catalog."default" NOT NULL,
    quantity_product real NOT NULL,
    required_stock real NOT NULL,
    type_product character varying(15) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT product_pkey PRIMARY KEY (id_product),
    CONSTRAINT product_quantity_product_check CHECK (quantity_product >= 0::double precision),
    CONSTRAINT product_id_product_check CHECK (id_product >= 0),
    CONSTRAINT product_required_stock_check CHECK (required_stock >= 0::double precision),
    CONSTRAINT product_id_product_check1 CHECK (id_product >= 0)
)
```

```
ALTER TABLE "Schema".product
    OWNER to postgres;

-- Создание таблицы provider и определение ограничений
CREATE TABLE "Schema".provider
(
    id_provider integer NOT NULL,
    name_provider character varying(50) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT provider_pkey PRIMARY KEY (id_provider),
    CONSTRAINT provider_id_provider_check CHECK (id_provider >= 0)
)
ALTER TABLE "Schema".provider
    OWNER to postgres;

-- Создание таблицы purchase и определение ограничений
CREATE TABLE "Schema".purchase
(
    id_purchase integer NOT NULL,
    date_purchase date NOT NULL,
    id_provider integer NOT NULL,
    CONSTRAINT purchase_pkey PRIMARY KEY (id_purchase),
    CONSTRAINT purchase_id_provider_fkey FOREIGN KEY (id_provider)
        REFERENCES "Schema".provider (id_provider) MATCH SIMPLE
        ON UPDATE RESTRICT
        ON DELETE RESTRICT,
    CONSTRAINT purchase_id_purchase_check CHECK (id_purchase >= 0)
)
ALTER TABLE "Schema".purchase
```

```
    OWNER to postgres;

-- Создание таблицы set of dishes и определение ограничений
CREATE TABLE "Schema"."set of dishes"
(
    id_set_dish integer NOT NULL,
    id_cook integer NOT NULL,
    id_dish integer NOT NULL,
    CONSTRAINT "set of dishes_pkey" PRIMARY KEY (id_set_dish),
    CONSTRAINT "set of dishes_id_cook_fkey" FOREIGN KEY (id_cook)
        REFERENCES "Schema".cooks (id_cook) MATCH SIMPLE
        ON UPDATE RESTRICT
        ON DELETE RESTRICT,
    CONSTRAINT "set of dishes_id_dish_fkey" FOREIGN KEY (id_dish)
        REFERENCES "Schema".dish (id_dish) MATCH SIMPLE
        ON UPDATE RESTRICT
        ON DELETE RESTRICT,
    CONSTRAINT "set of dishes_id_set_dish_check" CHECK (id_set_dish >= 0)
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE "Schema"."set of dishes"
    OWNER to postgres;

-- Создание таблицы structure of order и определение ограничений
```

```
CREATE TABLE "Schema"."structure of order"
(
    id_structure_order integer NOT NULL,
    id_order integer NOT NULL,
    id_dish integer NOT NULL,
    CONSTRAINT "structure of order_pkey" PRIMARY KEY (id_structure_order),
    CONSTRAINT "structure of order_id_dish_fkey" FOREIGN KEY (id_dish)
        REFERENCES "Schema".dish (id_dish) MATCH SIMPLE
        ON UPDATE RESTRICT
        ON DELETE RESTRICT,
    CONSTRAINT "structure of order_id_order_fkey" FOREIGN KEY (id_order)
        REFERENCES "Schema"."order" (id_order) MATCH SIMPLE
        ON UPDATE RESTRICT
        ON DELETE RESTRICT,
    CONSTRAINT "structure of order_id_structure_order_check" CHECK (id_structure_order >= 0)
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE "Schema"."structure of order"
    OWNER to postgres;

-- Создание таблицы table и определение ограничений
CREATE TABLE "Schema"."table"
(
```

```
    number_table integer NOT NULL,
    "info_table " character varying(20) COLLATE pg_catalog."default",
    id_order integer NOT NULL,
    CONSTRAINT table_pkey PRIMARY KEY (number_table),
    CONSTRAINT table_id_order_fkey FOREIGN KEY (id_order)
        REFERENCES "Schema"."order" (id_order) MATCH SIMPLE
        ON UPDATE RESTRICT
        ON DELETE RESTRICT,
    CONSTRAINT table_number_table_check CHECK (number_table >= 0)
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE "Schema"."table"
    OWNER to postgres;

-- Создание таблицы waiter и определение ограничений
CREATE TABLE "Schema".waiter
(
    id_waiter integer NOT NULL,
    id_worker integer NOT NULL,
    CONSTRAINT waiter_pkey PRIMARY KEY (id_waiter),
    CONSTRAINT waiter_id_worker_fkey FOREIGN KEY (id_worker)
        REFERENCES "Schema".worker (id_worker) MATCH SIMPLE
        ON UPDATE RESTRICT
        ON DELETE RESTRICT,
```

```
    CONSTRAINT waiter_id_waiter_check CHECK (id_waiter >= 0)
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE "Schema".waiter
    OWNER to postgres;

-- Создание таблицы worker и определение ограничений
CREATE TABLE "Schema".worker
(
    id_worker integer NOT NULL,
    name_worker character varying(60) COLLATE pg_catalog."default" NOT
NULL,
    passport_worker character varying(50) COLLATE pg_catalog."default" NOT
NULL,
    "id_possition " integer NOT NULL,
    CONSTRAINT worker_pkey PRIMARY KEY (id_worker),
    CONSTRAINT "worker_id_possition _fkey" FOREIGN KEY ("id_possition
")
        REFERENCES "Schema"."position" (id_position) MATCH SIMPLE
        ON UPDATE RESTRICT
        ON DELETE RESTRICT,
    CONSTRAINT worker_id_worker_check CHECK (id_worker >= 0)
)
WITH (
    OIDS = FALSE
)
```

```
  TABLESPACE pg_default;


  ALTER TABLE "Schema".worker
      OWNER to postgres;


--Заполнение таблицы composition of the purchase рабочими данными

INSERT INTO "Schema"."composition of the purchase"(
        id_structure_purchase, cost_product, quantity_product, shelf_life_product,
id_product, id_purchase)
        VALUES (1, 1000, 50, 2020-05-15, 1, 1), (2, 1500, 50, 2020-05-25, 2, 2),
(3, 2000, 50, 2020-05-25, 3, 3) ;

--Заполнение таблицы product рабочими данными

INSERT INTO "Schema".product(
        id_product, name_product, quantity_product, required_stock, type_product)
        VALUES (1, 'овощи', 50, 50, 'для гарнира'),  (2, 'фрукты', 50, 50, 'для
салата'),(3,'мясо',50, 50, 'горячее блюдо');

--Заполнение таблицы purchase рабочими данными

INSERT INTO "Schema".purchase(
        id_purchase, date_purchase, id_provider)
        VALUES (1, 2020-05-05, 1), (2, 2020-04-05, 2), (3, 2020-03-05, 3);

--Заполнение таблицы provider рабочими данными

INSERT INTO "Schema".provider(
        id_provider, name_provider)
        VALUES (1, 'Korovin индастрис'), (2, 'Рамкорм'), (3, 'фрутоняня');

--Заполнение таблицы cooks рабочими данными

INSERT INTO "Schema".cooks(
        id_cook, id_worker)
        VALUES (1, 1), (2, 2), (3, 3);

--Заполнение таблицы worker рабочими данными

INSERT INTO "Schema".worker(
        id_worker, name_worker, passport_worker, id_possition)
```

```
	VALUES (1, 'Виктор', 1000000000, 1), (2, 'Юрий', 2000000000, 2), (3,
'Александр', 3000000000, 3);

--Заполнение таблицы position рабочими данными

INSERT INTO "Schema"."position"(
	id_position, category_worker, salary)
	VALUES (1, 'официант', 20000), (2, 'директор', 50000), (3, 'повар',
35000);

--Заполнение таблицы waiter рабочими данными

INSERT INTO "Schema".waiter(
	id_waiter, id_worker)
	VALUES (1, 1),(2, 2),(3, 3)

--Заполнение таблицы table рабочими данными

INSERT INTO "Schema"."table"(
	number_table, "info_table ", id_order)
	VALUES (1, 'забронирован', 1),(2, 'свободен', 2),(3, 'свободен', 3)

--Заполнение таблицы order рабочими данными

INSERT INTO "Schema"."order"(
	id_order, date_order, number_table, stasus, id_waiter)
	VALUES (1, 2021-06-15, 1, 'занят', 1),(2, 2021-06-20, 2, 'свободен', 2),(3,
2021-06-10, 3, 'свободен', 3)

--Заполнение таблицы dish рабочими данными

INSERT INTO "Schema".dish(
	id_dish, name_dish, type_dish)
	VALUES (1, 'карбонара', 'горячее'), (2, 'пицца', 'горячее'), (3, 'стейк',
'горячее');

--Заполнение таблицы food ingredients рабочими данными

INSERT INTO "Schema"."food ingredients"(
	id_component, id_dish, id_product, quantity_ingredient_dish)
	VALUES (1, 1, 1, 50), (2, 2, 2, 50), (3, 3, 3, 50);

--Заполнение таблицы set of dishes рабочими данными

INSERT INTO "Schema"."set of dishes"(
```

```
        id_set_dish, id_cook, id_dish)
        VALUES (1, 1, 1), (2, 2, 2), (3, 3, 3);

--Заполнение таблицы structure of order рабочими данными

INSERT INTO "Schema"."structure of order"(
        id_structure_order, id_order, id_dish)
        VALUES (1, 1, 1), (2, 2, 2), (3, 3, 3);
```

Вывод:

В ходе выполнения работы была создана база данных в PostgreSQL, созданытаблицы и ограничения на значение столбцов, в базу данных были занесенырабочие данные, а также была создана логическая модель базы данных и dump.