

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**

по дисциплине: Проектирование и реализация баз данных

Специальность:

45.03.04 Интеллектуальные системы в гуманитарной сфере

Проверил:

Горова М.М. _____

Дата: «__» _____ 20__ г.

Оценка _____

Выполнила:

студентка

группы К3242

Редичкина А.М

Санкт-Петербург 2021

ЦЕЛЬ РАБОТЫ

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

ПРАКТИЧЕСКОЕ ЗАДАНИЕ

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.

ОСНОВНАЯ ЧАСТЬ

Вариант 13. БД «Ресторан»

Описание предметной области: Сотрудники ресторана – повара и официанты. За каждым официантом закреплены определенные столы. Каждый повар готовит определенный набор блюд. Запас продуктов на складе не должен быть ниже заданного значения. Цена заказа складывается из стоимости ингредиентов и наценки, которая составляет 40% стоимости ингредиентов.

БД должна содержать следующий минимальный набор сведений: ФИО сотрудника. Паспортные данные сотрудника. Категория сотрудника. Должность сотрудника. Оклад сотрудника. Наименование ингредиента. Код ингредиента. Дата закупки. Объем закупки. Количество продукта на складе. Необходимый запас продукта. Срок годности. Цена ингредиента. Поставщик. Наименование блюда. Код блюда. Объем ингредиента. Номер стола. Дата заказа. Код заказа. Количество. Название блюда. Ингредиенты, входящие в блюдо. Тип ингредиента.

1. Название БД

БД 'restaurant'

2. Схема инфологической модели данных БД

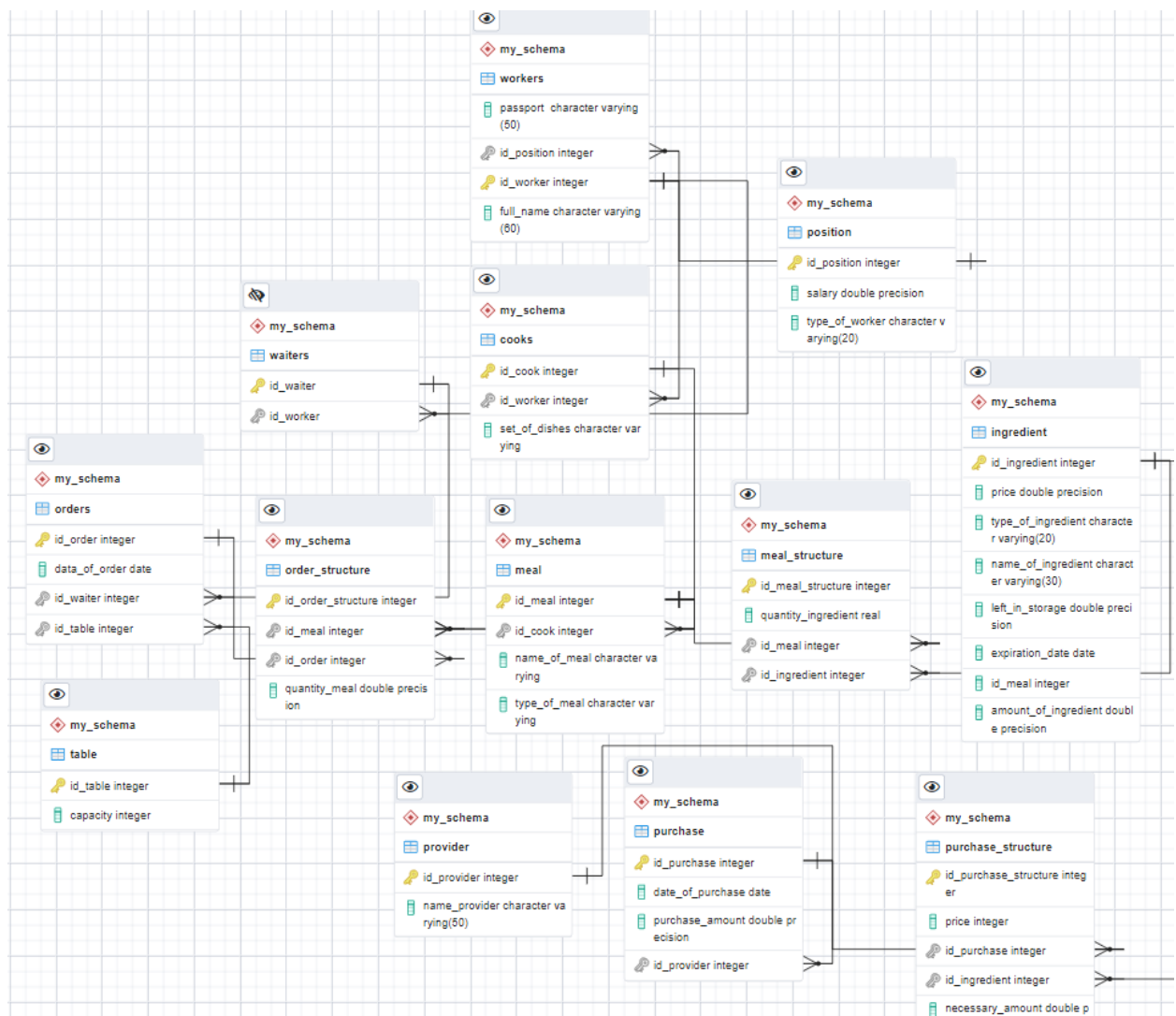


Рисунок 1 - Схема инфологической модели данных БД, сделанная в Generate GRE

3. Создание базы данных и ее таблиц. Установление ограничений на данные.

1) Создание базы данных "restaurant":

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

2) Создание таблицы “workers”:

```
CREATE TABLE my_schema.workers
(
    "passport " character varying(50) COLLATE pg_catalog."default" NOT NULL,
    id_position integer NOT NULL,
    id_worker integer NOT NULL,
    full_name character varying(60) COLLATE pg_catalog."default",
    CONSTRAINT id_worker PRIMARY KEY (id_worker),
    CONSTRAINT id_position FOREIGN KEY (id_position)
        REFERENCES my_schema."position" (id_position) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)
```

```
TABLESPACE pg_default;
```

```
ALTER TABLE my_schema.workers
    OWNER to postgres;
```

```
CREATE INDEX fki_id_position
    ON my_schema.workers USING btree
    (id_position ASC NULLS LAST)
    TABLESPACE pg_default;
```

3) Создание таблицы “position”:

```
CREATE TABLE my_schema."position"
(
    id_position integer NOT NULL,
    salary double precision,
    type_of_worker character varying COLLATE pg_catalog."default",
    CONSTRAINT position_pkey PRIMARY KEY (id_position),
    CONSTRAINT type_of_worker CHECK (type_of_worker::text = ANY
    (ARRAY['официант'::character varying::text, 'повар'::character varying::text,
```

```
'шеф-повар'::character varying::text, 'администратор'::character varying::text])) NOT  
VALID  
)
```

```
TABLESPACE pg_default;
```

```
ALTER TABLE my_schema."position"  
OWNER to postgres;
```

4) Создание таблицы “waiters”:

```
CREATE TABLE my_schema.waiters  
(  
    id_waiter integer NOT NULL,  
    id_worker integer,  
    CONSTRAINT waiters_pkey PRIMARY KEY (id_waiter),  
    CONSTRAINT id_worker FOREIGN KEY (id_worker)  
        REFERENCES my_schema.workers (id_worker) MATCH SIMPLE  
        ON UPDATE NO ACTION  
        ON DELETE NO ACTION  
        NOT VALID  
)
```

```
TABLESPACE pg_default;
```

```
ALTER TABLE my_schema.waiters  
OWNER to postgres;
```

```
CREATE INDEX fki_id_worker  
ON my_schema.waiters USING btree  
(id_worker ASC NULLS LAST)  
TABLESPACE pg_default;
```

5) Создание таблицы “orders”

```
CREATE TABLE my_schema.orders  
(  
    id_order integer NOT NULL,  
    data_of_order date,  
    id_waiter integer,
```

```

id_table integer,
CONSTRAINT "Orders_pkey" PRIMARY KEY (id_order),
CONSTRAINT id_table FOREIGN KEY (id_table)
    REFERENCES my_schema."table" (id_table) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID,
CONSTRAINT id_waiter FOREIGN KEY (id_waiter)
    REFERENCES my_schema.waiters (id_waiter) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID
)

```

```

TABLESPACE pg_default;

```

```

ALTER TABLE my_schema.orders
    OWNER to postgres;

```

```

CREATE INDEX fki_id_table
    ON my_schema.orders USING btree
    (id_table ASC NULLS LAST)
    TABLESPACE pg_default;

```

```

CREATE INDEX fki_id_waiter
    ON my_schema.orders USING btree
    (id_waiter ASC NULLS LAST)
    TABLESPACE pg_default;

```

6) Создание таблицы “cooks”:

```

CREATE TABLE my_schema.cooks
(
    id_cook integer NOT NULL,
    id_worker integer NOT NULL,
    set_of_dishes character varying COLLATE pg_catalog."default",
    CONSTRAINT cooks_pkey PRIMARY KEY (id_cook),
    CONSTRAINT id_worker FOREIGN KEY (id_worker)
        REFERENCES my_schema.workers (id_worker) MATCH SIMPLE

```

```
ON UPDATE NO ACTION
ON DELETE NO ACTION
NOT VALID
)
```

```
TABLESPACE pg_default;
```

```
ALTER TABLE my_schema.cooks
OWNER to postgres;
```

7) Создание таблицы “meal”:

```
CREATE TABLE my_schema.meal
(
    id_meal integer NOT NULL,
    id_cook integer,
    name_of_meal character varying COLLATE pg_catalog."default" NOT NULL,
    type_of_meal character varying COLLATE pg_catalog."default",
    CONSTRAINT meal_pkey PRIMARY KEY (id_meal),
    CONSTRAINT id_cook FOREIGN KEY (id_cook)
        REFERENCES my_schema.cooks (id_cook) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT type_of_meal CHECK (type_of_meal::text = ANY
(ARRAY['суп'::character varying::text, 'горячее блюдо'::character varying::text,
'салат'::character varying::text, 'напиток'::character varying::text, 'десерт'::character
varying::text, 'закуска'::character varying::text]))
)
```

```
TABLESPACE pg_default;
```

```
ALTER TABLE my_schema.meal
OWNER to postgres;
```

```
CREATE INDEX fki_id_cook
ON my_schema.meal USING btree
(id_cook ASC NULLS LAST)
TABLESPACE pg_default;
```

8) Создание таблицы "table":

```
CREATE TABLE my_schema."table"  
(  
    id_table integer NOT NULL,  
    capacity integer,  
    CONSTRAINT table_pkey PRIMARY KEY (id_table),  
    CONSTRAINT amount_of_tables CHECK (id_table < 30),  
    CONSTRAINT capacity CHECK (capacity <= 10)  
)
```

```
TABLESPACE pg_default;
```

```
ALTER TABLE my_schema."table"  
    OWNER to postgres;
```

9) Создание таблицы "purchase":

```
CREATE TABLE my_schema.purchase  
(  
    id_purchase integer NOT NULL,  
    date_of_purchase date,  
    purchase_amount double precision,  
    id_provider integer NOT NULL,  
    CONSTRAINT purchase_pkey PRIMARY KEY (id_purchase),  
    CONSTRAINT id_provider FOREIGN KEY (id_provider)  
        REFERENCES my_schema.provider (id_provider) MATCH SIMPLE  
        ON UPDATE NO ACTION  
        ON DELETE NO ACTION  
        NOT VALID  
)
```

```
TABLESPACE pg_default;
```

```
ALTER TABLE my_schema.purchase  
    OWNER to postgres;
```

```
CREATE INDEX fki_id_provider  
    ON my_schema.purchase USING btree
```



```
(id_provider ASC NULLS LAST)
TABLESPACE pg_default;
```

10) Создание таблицы “purchase_structure”:

```
CREATE TABLE my_schema.purchase_structure
(
    id_purchase_structure integer NOT NULL,
    price integer,
    id_purchase integer,
    id_ingredient integer,
    necessary_amount double precision,
    CONSTRAINT purchase_structure_pkey PRIMARY KEY (id_purchase_structure),
    CONSTRAINT id_ingredient FOREIGN KEY (id_ingredient)
        REFERENCES my_schema.ingredient (id_ingredient) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT id_purhase FOREIGN KEY (id_purchase)
        REFERENCES my_schema.purchase (id_purchase) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)
```

```
TABLESPACE pg_default;
```

```
ALTER TABLE my_schema.purchase_structure
    OWNER to postgres;
```

```
CREATE INDEX fki_id_ingredient
    ON my_schema.purchase_structure USING btree
    (id_ingredient ASC NULLS LAST)
    TABLESPACE pg_default;
```

```
CREATE INDEX fki_id_purhase
    ON my_schema.purchase_structure USING btree
```

```
(id_purchase ASC NULLS LAST)
TABLESPACE pg_default;
```

11) Создание таблицы “provider”:

```
CREATE TABLE my_schema.provider
(
    id_provider integer NOT NULL,
    name_provider character varying(50) COLLATE pg_catalog."default",
    CONSTRAINT provider_pkey PRIMARY KEY (id_provider)
)
```

```
TABLESPACE pg_default;
```

```
ALTER TABLE my_schema.provider
    OWNER to postgres;
```

12) Создание таблицы “ingredients”:

```
CREATE TABLE my_schema.ingredient
(
    id_ingredient integer NOT NULL,
    price double precision,
    type_of_ingredient character varying(20) COLLATE pg_catalog."default",
    name_of_ingredient character varying(30) COLLATE pg_catalog."default",
    left_in_storage double precision,
    expiration_date date,
    id_meal integer,
    amount_of_ingredient double precision,
    CONSTRAINT ingredient_pkey PRIMARY KEY (id_ingredient),
    CONSTRAINT type_of_ingredient CHECK (type_of_ingredient::text = ANY
(ARRAY['мясо'::character varying::text, 'овощи'::character varying::text, 'фрукты'::character
varying::text, 'рыба'::character varying::text, 'молочный продукт'::character varying::text,
'приправа'::character varying::text, 'соус'::character varying::text, 'яйца'::character
varying::text, 'бобы'::character varying::text]))
)
```

```
TABLESPACE pg_default;
```

```
ALTER TABLE my_schema.ingredient
```

OWNER to postgres;

```
CREATE INDEX fki_id_meal
ON my_schema.ingredient USING btree
(id_meal ASC NULLS LAST)
TABLESPACE pg_default;
```

13)Создание таблицы “meal_structure”:

```
CREATE TABLE my_schema.meal_structure
(
    id_meal_structure integer NOT NULL,
    quantity_ingredient real,
    id_meal integer,
    id_ingredient integer,
    CONSTRAINT meal_structure_pkey PRIMARY KEY (id_meal_structure),
    CONSTRAINT id_ing_in_structure FOREIGN KEY (id_meal)
        REFERENCES my_schema.meal (id_meal) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT id_ingredient_in_meal FOREIGN KEY (id_ingredient)
        REFERENCES my_schema.ingredient (id_ingredient) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)
```

TABLESPACE pg_default;

```
ALTER TABLE my_schema.meal_structure
OWNER to postgres;
```

```
CREATE INDEX fki_id_ing_in_structure
ON my_schema.meal_structure USING btree
(id_meal ASC NULLS LAST)
TABLESPACE pg_default;
```

```
CREATE INDEX fki_id_ingredient_in_meal
ON my_schema.meal_structure USING btree
```

```
(id_ingredient ASC NULLS LAST)
TABLESPACE pg_default;
```

14)Создание таблицы “order_sctructure”:

```
CREATE TABLE my_schema.order_structure
(
    id_order_structure integer NOT NULL,
    id_meal integer,
    id_order integer,
    quantity_meal double precision,
    CONSTRAINT order_structure_pkey PRIMARY KEY (id_order_structure),
    CONSTRAINT id_meal FOREIGN KEY (id_meal)
        REFERENCES my_schema.meal (id_meal) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT id_meal_in_order FOREIGN KEY (id_meal)
        REFERENCES my_schema.meal (id_meal) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT id_struct_order FOREIGN KEY (id_order)
        REFERENCES my_schema.orders (id_order) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)
```

```
TABLESPACE pg_default;
```

```
ALTER TABLE my_schema.order_structure
    OWNER to postgres;
```

```
CREATE INDEX fki_id_meal_in_order
    ON my_schema.order_structure USING btree
    (id_meal ASC NULLS LAST)
    TABLESPACE pg_default;
```

```
CREATE INDEX fki_id_struct_order
```

```
ON my_schema.order_structure USING btree
(id_order ASC NULLS LAST)
TABLESPACE pg_default;
```

4. Заполнение таблицы БД рабочими данными

1) Заполнение данными таблицы “workers”:

```
INSERT INTO my_schema.workers(
    full_name, "passport ", id_position, id_worker)
VALUES ('Иванов Алексей Дмитриевич', '1217 567831', 2, 1),
('Банкеева Диана Андреевна', '1615 354821', 2, 2),
('Маркова Элина Максимовна', '1216 545761', 5, 3),
('Корнев Арсентий Валерьевич', '1715 345767', 4, 4),
('Аксёнова Юлия Константиновна', '1427 813475', 3, 5),
('Яровой Максим Владимирович', '1356 797231', 3, 6);
```

Результат:





	 passport character varying (50)	 id_position integer	 id_worker [PK] integer	 full_name character varying (60)
1	1217 567831	2	1	Иванов Алексей Дмитриевич
2	1615 354821	2	2	Банкеева Диана Андреевна
3	1216 545761	5	3	Маркова Элина Максимовна
4	1715 345767	4	4	Корнев Арсентий Валерьевич
5	1427 813475	3	5	Аксёнова Юлия Константиновна
6	1356 797231	3	6	Яровой Максим Владимирович

Рисунок 2 - рабочие данные для таблицы ‘workers’

2) Заполнение данными таблицы “cooks”:

```
INSERT INTO my_schema.cooks(
    id_cook, id_worker, set_of_dishes)
VALUES (1, 3, 'супы, горячие блюда, салаты, закуски'),
(2, 5, 'салаты, закуски'),
(3, 6, 'супы, горячие блюда');
```

Результат:





Результат		План выполнения		Сообщения	Notifications
	id_cook [PK] integer 	id_worker integer 	set_of_dishes character varying 		
1	1	3	супы, горячие блюда, салаты, закуски		
2	2	5	салаты, закуски		
3	3	6	супы, горячие блюда		

Рисунок 3 - рабочие данные для таблицы 'cooks'

3) Заполнение данными таблицы "position":

```
INSERT INTO my_schema.position(  
    id_position, salary, type_of_worker)  
VALUES (2, 25000, 'официант'),  
       (3, 60000, 'повар'),  
       (4, 50000, 'администратор'),  
       (5, 70000, 'шеф-повар');
```

Результат:

	<div>id_position</div> <div>[PK] integer</div>	<div>salary</div> <div>double precision</div>	<div>type_of_worker</div> <div>character varying (20)</div>
1	2	25000	официант
2	3	60000	повар
3	4	50000	администратор
4	5	70000	шеф-повар

Рисунок 4 - рабочие данные для таблицы 'position'

4) Заполнение данными таблицы "waiters":

```
INSERT INTO my_schema.waiters(  
    id_waiter, id_worker)
```

```
VALUES (1, 1),  
(2,2);
```

Результат:



	 id_waiter [PK] integer	 id_worker integer
1	1	1
2	2	2

Рисунок 5 - рабочие данные для таблицы 'waiters'

5) Заполнение данными таблицы 'table':

```
INSERT INTO my_schema."table"(  
    id_table, capacity)  
VALUES (1, 8),  
(2, 4),  
(3, 2),  
(4, 6),  
(5, 6),  
(6, 8),  
(7, 4),  
(8, 4),  
(9, 2),  
(10, 4);
```

Результат:

	id_table [PK] integer	capacity integer
1	1	8
2	2	4
3	3	2
4	4	6
5	5	6
6	6	8
7	7	4
8	8	4
9	9	2
10	10	4

Рисунок 6 - рабочие данные для таблицы 'table'

6) Заполнение данными таблицы "orders":

```
INSERT INTO my_schema.orders(
  id_order, data_of_order, id_waiter, id_table)
VALUES (1, '2021-04-09', 1, 4),
(2, '2021-04-09', 2, 5),
(3, '2021-04-09', 1, 2),
(4, '2021-04-10', 2, 6),
(5, '2021-04-10', 2, 7),
(6, '2021-04-10', 1, 10),
(7, '2021-04-11', 1, 2),
(8, '2021-04-11', 1, 4),
(9, '2021-04-11', 2, 9);
```

Результат:






	 id_order [PK] integer 	data_of_order date 	id_waiter integer 	id_table integer 
1	1	2021-04-09	1	4
2	2	2021-04-09	2	5
3	3	2021-04-09	1	2
4	4	2021-04-10	2	6
5	5	2021-04-10	2	7
6	6	2021-04-10	1	10
7	7	2021-04-11	1	2
8	8	2021-04-11	1	4
9	9	2021-04-11	2	9

Рисунок 7- рабочие данные для таблицы 'orders'

Вывод:

В результате выполнения работы в программе pgadmin4 была создана база данных 'restaurant', таблицы. Были наложены ограничения на данные, а также выполнена вставка рабочих данных в таблицы.