Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
(Университет ИТМО)

Факультет **Инфокоммуникационных технологий**

Образовательная программа **Мобильные и сетевые технологии**

Направление подготовки **09.03.03 Прикладная информатика**

О Т Ч Е Т

Лабораторная работа 3

Тема: Создание таблиц базы данных PostgreSQL. Заполнение таблиц рабочими данными.

Обучающийся: Файзулин Радмир Русланович, группы К3239

Преподаватель: Говорова Марина Михайловна

Санкт-Петербург 2024

СОДЕРЖАНИЕ

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 (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:

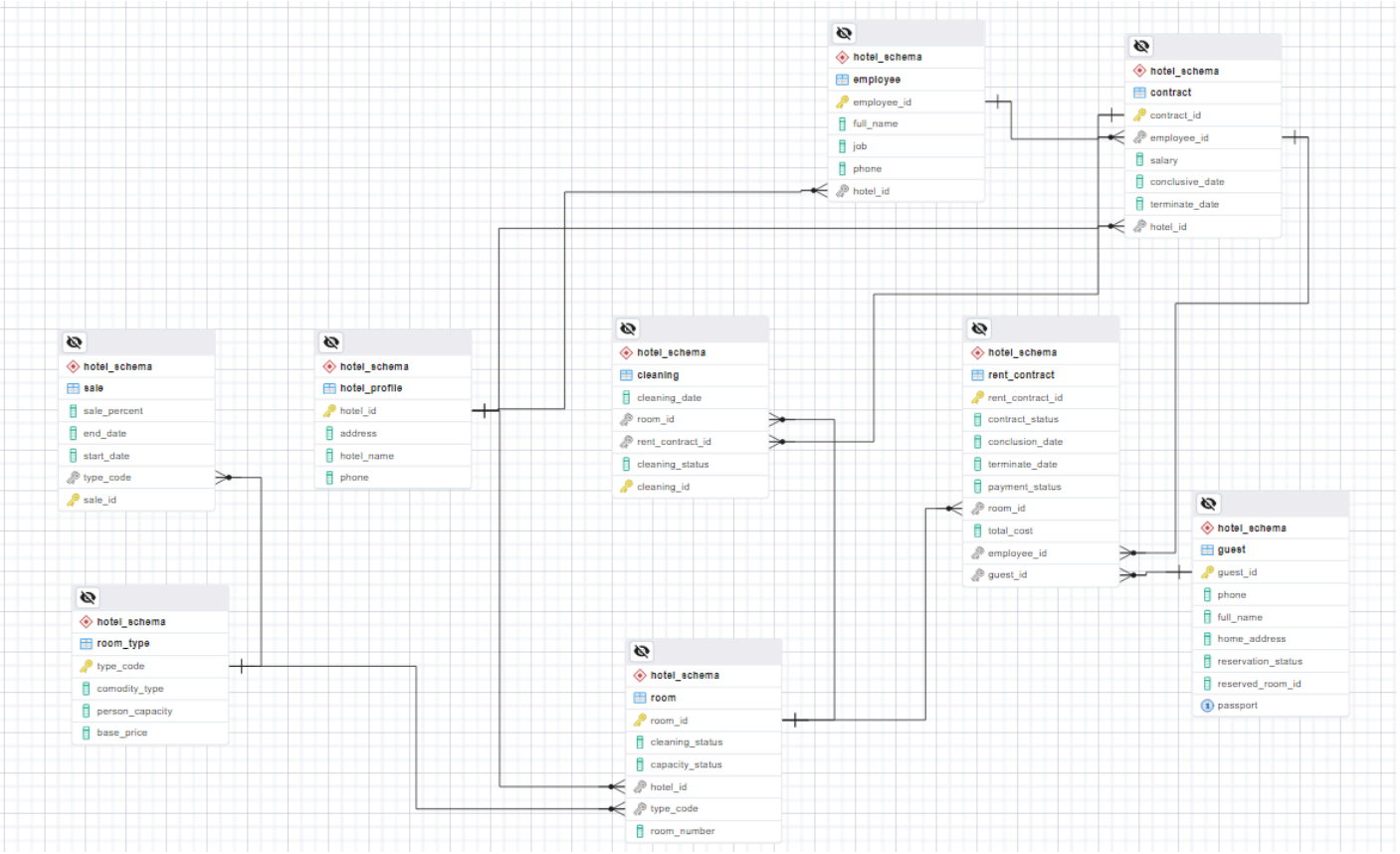
Создать две резервные копии:

- *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.

Ход работы

Схема базы данных



Hotel (Отель)

Далее скрипты таблиц

```
-- Table: hotel_schema.room
```

```
-- DROP TABLE IF EXISTS hotel_schema.room;

CREATE TABLE IF NOT EXISTS hotel_schema.room
(
    room_id integer NOT NULL GENERATED BY DEFAULT AS IDENTITY ( INCREMENT
1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),
    cleaning_status character varying(20) COLLATE pg_catalog."default" NOT
NULL,
    capacity_status character varying(20) COLLATE pg_catalog."default" NOT
NULL,
    hotel_id integer NOT NULL,
    type_code integer NOT NULL,
```

```
    room_number integer,
    CONSTRAINT room_pk PRIMARY KEY (room_id),
    CONSTRAINT room_hotel_profile_hotel_id_fk FOREIGN KEY (hotel_id)
        REFERENCES hotel_schema.hotel_profile (hotel_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT room_room_type_type_code_fk FOREIGN KEY (type_code)
        REFERENCES hotel_schema.room_type (type_code) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT check_room_id CHECK (room_id > 0 AND room_id <= 2000),
     CONSTRAINT check_capacity_status CHECK (capacity_status::text = ANY
(ARRAY['free'::character varying, 'busy'::character varying]::text[])),
    CONSTRAINT check_hotel_id CHECK (hotel_id >= 1 AND hotel_id <= 5),
     CONSTRAINT check_cleaning_status CHECK (cleaning_status::text = 'to
do'::text OR cleaning_status::text = 'done'::text)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS hotel_schema.room
    OWNER to postgres;
```

```
-- Table: hotel_schema.guest

-- DROP TABLE IF EXISTS hotel_schema.guest;

CREATE TABLE IF NOT EXISTS hotel_schema.guest
(
    guest_id integer NOT NULL GENERATED BY DEFAULT AS IDENTITY ( INCREMENT
1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),
    phone character varying(20) COLLATE pg_catalog."default",
      full_name character varying(100) COLLATE pg_catalog."default" NOT
NULL,
     home_address character varying(70) COLLATE pg_catalog."default" NOT
NULL,
     reservation_status character varying(20) COLLATE pg_catalog."default"
NOT NULL,
    reserved_room_id integer NOT NULL,
    passport character varying(20) COLLATE pg_catalog."default",
    CONSTRAINT guest_pk PRIMARY KEY (guest_id),
    CONSTRAINT passport_uk UNIQUE (passport),
```

```
    CONSTRAINT check_reservation_status CHECK (reservation_status::text =
'reserves'::text   OR   reservation_status::text   =   'rents'::text   OR
reservation_status::text = 'left'::text),
    CONSTRAINT check_phone CHECK (phone::text ~ '^[0-9]+$'::text),
          CONSTRAINT   check_guest_full_name   CHECK   (full_name::text   ~
'^[A-Za-zA-ЯЁa-яё -]+$'::text),
          CONSTRAINT   check_home_address   CHECK   (home_address::text   ~
'^[A-Za-zA-ЯЁa-яё0-9.,/ -]+$'::text),
    CONSTRAINT check_passport CHECK (passport::text ~ '^[0-9 ]+$'::text)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS hotel_schema.guest
    OWNER to postgres;
```

```
-- Table: hotel_schema.hotel_profile

-- DROP TABLE IF EXISTS hotel_schema.hotel_profile;

CREATE TABLE IF NOT EXISTS hotel_schema.hotel_profile
(
    hotel_id integer NOT NULL GENERATED BY DEFAULT AS IDENTITY ( INCREMENT
1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),
    address character varying(100) COLLATE pg_catalog."default" NOT NULL,
      hotel_name  character  varying(20)  COLLATE  pg_catalog."default"  NOT
NULL,
    phone character varying(20) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT hotel_profile_pk PRIMARY KEY (hotel_id),
    CONSTRAINT check_hotel_id CHECK (hotel_id > 0),
    CONSTRAINT check_phone CHECK (phone::text ~ '^[0-9]+$'::text),
               CONSTRAINT    check_address    CHECK    (address::text    ~
'^[A-Za-zA-ЯЁa-яё0-9/., -]+$'::text)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS hotel_schema.hotel_profile
    OWNER to postgres;
```

```
-- Table: hotel_schema.rent_contract

-- DROP TABLE IF EXISTS hotel_schema.rent_contract;

CREATE TABLE IF NOT EXISTS hotel_schema.rent_contract
(
    rent_contract_id integer NOT NULL GENERATED BY DEFAULT AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),
    contract_status character varying(20) COLLATE pg_catalog."default" NOT NULL,
    conclusion_date date NOT NULL,
    terminate_date date NOT NULL,
    payment_status character varying(20) COLLATE pg_catalog."default" NOT NULL,
    room_id integer NOT NULL,
    total_cost bigint,
    employee_id integer NOT NULL,
    guest_id integer NOT NULL,
    CONSTRAINT rent_contract_pk_3 PRIMARY KEY (rent_contract_id),
    CONSTRAINT contract_guest_passport_id_fk FOREIGN KEY (guest_id)
        REFERENCES hotel_schema.guest (guest_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
        CONSTRAINT contract_rent_contract_employee_id_fk FOREIGN KEY (employee_id)
        REFERENCES hotel_schema.contract (employee_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT contract_room_room_id_fk FOREIGN KEY (room_id)
        REFERENCES hotel_schema.room (room_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT check_contract_status CHECK (contract_status::text = ANY (ARRAY['valid'::character varying, 'finished'::character varying, 'reserving'::character varying]::text[])),
    CONSTRAINT check_payment_status CHECK (payment_status::text = ANY (ARRAY['paid'::character varying, 'not paid'::character varying]::text[])),
    CONSTRAINT check_dates CHECK (conclusion_date <= terminate_date),
    CONSTRAINT check_total_cost CHECK (total_cost > 0 AND total_cost <= 1000000000)
```

```
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS hotel_schema.rent_contract
    OWNER to postgres;
```

```
-- Table: hotel_schema.contract

-- DROP TABLE IF EXISTS hotel_schema.contract;

CREATE TABLE IF NOT EXISTS hotel_schema.contract
(
      contract_id integer NOT NULL GENERATED BY DEFAULT AS IDENTITY (
INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),
    employee_id integer NOT NULL,
    salary integer NOT NULL,
    conclusive_date date NOT NULL,
    terminate_date date,
    hotel_id integer NOT NULL,
    CONSTRAINT contract_pk PRIMARY KEY (contract_id),
    CONSTRAINT contract_pk_3 UNIQUE (employee_id),
         CONSTRAINT  rent_contract_employee_employee_id_fk  FOREIGN  KEY
(employee_id)
        REFERENCES hotel_schema.employee (employee_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
         CONSTRAINT  rent_contract_hotel_profile_hotel_id_fk  FOREIGN  KEY
(hotel_id)
        REFERENCES hotel_schema.hotel_profile (hotel_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT check_salary CHECK (salary > 0),
    CONSTRAINT check_employee_id CHECK (employee_id > 0),
    CONSTRAINT check_dates CHECK (conclusive_date < terminate_date),
    CONSTRAINT check_hotel_id CHECK (hotel_id > 0 AND hotel_id < 6)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS hotel_schema.contract
    OWNER to postgres;
```

```
-- Table: hotel_schema.employee

-- DROP TABLE IF EXISTS hotel_schema.employee;

CREATE TABLE IF NOT EXISTS hotel_schema.employee
(
      employee_id  integer  NOT  NULL  GENERATED  BY  DEFAULT  AS  IDENTITY  (
INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),
    full_name character varying(70) COLLATE pg_catalog."default" NOT NULL,
    job character varying(70) COLLATE pg_catalog."default" NOT NULL,
    phone character varying(20) COLLATE pg_catalog."default" NOT NULL,
    hotel_id integer,
    CONSTRAINT employee_pk PRIMARY KEY (employee_id),
    CONSTRAINT employee_hotel_id_fk FOREIGN KEY (hotel_id)
        REFERENCES hotel_schema.hotel_profile (hotel_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
         CONSTRAINT  check_job  CHECK  (job::text  ~  '^[A-Za-zA-ЯЁa-яё0-9
-]+$'::text),
     CONSTRAINT check_full_name CHECK (full_name::text ~ '^[A-Za-zA-ЯЁa-яё
-]+$'::text),
    CONSTRAINT check_phone CHECK (phone::text ~ '^[0-9]+$'::text)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS hotel_schema.employee
    OWNER to postgres;
```

```
-- Table: hotel_schema.cleaning

-- DROP TABLE IF EXISTS hotel_schema.cleaning;

CREATE TABLE IF NOT EXISTS hotel_schema.cleaning
(
    cleaning_date date NOT NULL,
    room_id integer NOT NULL,
```

```
    rent_contract_id integer NOT NULL,
    cleaning_status character varying(20) COLLATE pg_catalog."default" NOT
NULL,
      cleaning_id integer NOT NULL GENERATED BY DEFAULT AS IDENTITY (
INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),
    CONSTRAINT cleaning_pk_4 PRIMARY KEY (cleaning_id),
    CONSTRAINT cleaning_room_room_id_fk FOREIGN KEY (room_id)
        REFERENCES hotel_schema.room (room_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
             CONSTRAINT   rent_contract_id_cleaning___fk   FOREIGN   KEY
(rent_contract_id)
        REFERENCES hotel_schema.contract (contract_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
            CONSTRAINT   check_cleaning_date   CHECK   (cleaning_date   >
'2020-01-01'::date),
     CONSTRAINT check_cleaning_status CHECK (cleaning_status::text = 'to
do'::text OR cleaning_status::text = 'done'::text),
    CONSTRAINT check_room_id CHECK (room_id > 0),
    CONSTRAINT check_rent_contract_id CHECK (rent_contract_id > 0)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS hotel_schema.cleaning
    OWNER to postgres;
```

Сама схема:

```
-- SCHEMA: hotel_schema
```

```
-- DROP SCHEMA IF EXISTS hotel_schema ;

CREATE SCHEMA IF NOT EXISTS hotel_schema
    AUTHORIZATION postgres;
```

Данные также сохранены по отдельности в общей папке.

Вывод:

Я научился создавать схему, таблицы и в целом базы данных, добавляя нужные ограничения, проверки. Затем вносить, изменять и удалять данные. Также вручную была сделана резервная копия базы данных.
Скрипты представлены выше!