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:
09.03.03 Мобильные и сетевые технологии

Проверил:
Говорова М.М. _____
Дата: «__» ____ 2021 г.
Оценка _____

Выполнил:
студент группы К3241

Касаткин Д.А.

Санкт-Петербург 2021 г.

ЦЕЛЬ РАБОТЫ

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

ПРАКТИЧЕСКОЕ ЗАДАНИЕ

Создать базу данных с использованием pgAdmin 4 (Вариант 8. БД «Аэропорт»).

1. Создать схему в составе базы данных.
2. Создать таблицы базы данных.
3. Установить ограничения на данные: *Primary Key, Unique, Check, Foreign Key*.
4. Заполнить таблицы БД рабочими данными.
5. Создать резервную копию БД.

Указание:

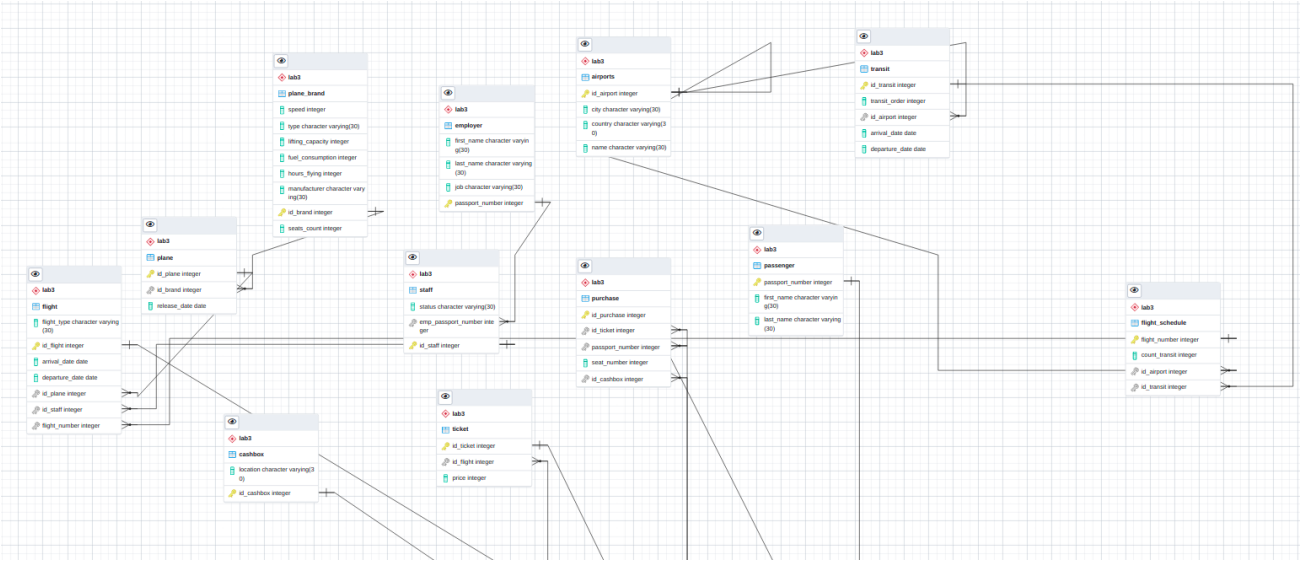
Создать две резервные копии:

- *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 .*

6. Восстановить БД.

ВЫПОЛНЕНИЕ

1. Наименование БД “postgres”
2. Схема логической модели базы данных, сгенерированная в Generate ERD



3. Создание базы данных

```
-- Database: postgres

-- DROP DATABASE postgres;

CREATE DATABASE postgres
    WITH
    OWNER = postgres
    ENCODING = 'UTF8'
    LC_COLLATE = 'en_US.UTF-8'
    LC_CTYPE = 'en_US.UTF-8'
    TABLESPACE = pg_default
    CONNECTION LIMIT = -1;

COMMENT ON DATABASE postgres
    IS 'default administrative connection database';
```

4. Создание схемы

```
-- SCHEMA: lab3

-- DROP SCHEMA lab3 ;

CREATE SCHEMA lab3
    AUTHORIZATION daniil;
```

5. Создание таблиц, колонок и ограничений

```
-- Table: lab3.airports

-- DROP TABLE lab3.airports;

CREATE TABLE IF NOT EXISTS lab3.airports
(
    id_airport integer NOT NULL,
    city character varying(30) COLLATE pg_catalog."default" NOT NULL,
    country character varying(30) COLLATE pg_catalog."default" NOT NULL,
    name character varying(30) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT airports_pkey PRIMARY KEY (id_airport)
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE lab3.airports
    OWNER to daniil;
```

```sql
-- Table: lab3.cashbox

-- DROP TABLE lab3.cashbox;

CREATE TABLE IF NOT EXISTS lab3.cashbox
(
    location character varying(30) COLLATE pg_catalog."default" NOT NULL,
    id_cashbox integer NOT NULL,
    CONSTRAINT cashbox_pkey PRIMARY KEY (id_cashbox)
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE lab3.cashbox
    OWNER to daniil;
```

```sql
-- Table: lab3.employer

-- DROP TABLE lab3.employer;

CREATE TABLE IF NOT EXISTS lab3.employer
(
    first_name character varying(30) COLLATE pg_catalog."default" NOT NULL,
    last_name character varying(30) COLLATE pg_catalog."default" NOT NULL,
    job character varying(30) COLLATE pg_catalog."default" NOT NULL,
    passport_number integer NOT NULL,
    CONSTRAINT employer_pkey PRIMARY KEY (passport_number),
    CONSTRAINT job_check CHECK (job::text = ANY (ARRAY['первый пилот'::character varying::text, 'второй пилот'::character varying::text, 'стюард'::character varying::text, 'стюардесса'::ch
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE lab3.employer
    OWNER to daniil;
```

Dashboard Properties SQL Statistics Dependencies Dependents postgres/daniil...

```sql
-- Table: lab3.flight

-- DROP TABLE lab3.flight;

CREATE TABLE IF NOT EXISTS lab3.flight
(
    flight_type character varying(30) COLLATE pg_catalog."default" NOT NULL,
    id_flight integer NOT NULL,
    arrival_date date NOT NULL,
    departure_date date NOT NULL,
    id_plane integer NOT NULL,
    id_staff integer NOT NULL,
    flight_number integer NOT NULL,
    CONSTRAINT flight_pkey PRIMARY KEY (id_flight),
    CONSTRAINT flight_id_plane_fkey FOREIGN KEY (id_plane)
        REFERENCES lab3.plane (id_plane) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT flight_id_staff_fkey FOREIGN KEY (id_staff)
        REFERENCES lab3.staff (id_staff) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT flight_number FOREIGN KEY (flight_number)
        REFERENCES lab3.flight_schedule (flight_number) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT departure_date_check CHECK (departure_date > '1900-01-01'::date) NOT VALID,
    CONSTRAINT arrival_date_check CHECK (arrival_date > departure_date) NOT VALID
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE lab3.flight
    OWNER to daniil;
-- Index: fki_flight_number

-- DROP INDEX lab3.fki_flight_number;

CREATE INDEX fki_flight_number
    ON lab3.flight USING btree
    (flight_number ASC NULLS LAST)
    TABLESPACE pg_default;
-- Index: fki_id_plane

-- DROP INDEX lab3.fki_id_plane;
```

```
    CONSTRAINT flight_pkey PRIMARY KEY (id_flight),
    CONSTRAINT flight_id_plane_fkey FOREIGN KEY (id_plane)
        REFERENCES lab3.plane (id_plane) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT flight_id_staff_fkey FOREIGN KEY (id_staff)
        REFERENCES lab3.staff (id_staff) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT flight_number FOREIGN KEY (flight_number)
        REFERENCES lab3.flight_schedule (flight_number) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT departure_date_check CHECK (departure_date > '1900-01-01'::date) NOT VALID,
    CONSTRAINT arrival_date_check CHECK (arrival_date > departure_date) NOT VALID
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE lab3.flight
    OWNER to daniil;
-- Index: fki_flight_number

-- DROP INDEX lab3.fki_flight_number;

CREATE INDEX fki_flight_number
    ON lab3.flight USING btree
    (flight_number ASC NULLS LAST)
    TABLESPACE pg_default;
-- Index: fki_id_plane

-- DROP INDEX lab3.fki_id_plane;

CREATE INDEX fki_id_plane
    ON lab3.flight USING btree
    (id_plane ASC NULLS LAST)
    TABLESPACE pg_default;
-- Index: fki_id_staff

-- DROP INDEX lab3.fki_id_staff;

CREATE INDEX fki_id_staff
    ON lab3.flight USING btree
    (id_staff ASC NULLS LAST)
    TABLESPACE pg_default;
```

```
-- Table: lab3.flight_schedule

-- DROP TABLE lab3.flight_schedule;

CREATE TABLE IF NOT EXISTS lab3.flight_schedule
(
    flight_number integer NOT NULL,
    count_transit integer NOT NULL,
    id_airport integer NOT NULL,
    id_transit integer NOT NULL,
    CONSTRAINT flight_schedule_pkey PRIMARY KEY (flight_number),
    CONSTRAINT flight_schedule_id_airport_fkey FOREIGN KEY (id_airport)
        REFERENCES lab3.airports (id_airport) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT flight_schedule_id_transit_fkey FOREIGN KEY (id_transit)
        REFERENCES lab3.transit (id_transit) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE lab3.flight_schedule
    OWNER to daniil;
-- Index: fki_id_airport

-- DROP INDEX lab3.fki_id_airport;

CREATE INDEX fki_id_airport
    ON lab3.flight_schedule USING btree
    (id_airport ASC NULLS LAST)
    TABLESPACE pg_default;
-- Index: fki_id_transit

-- DROP INDEX lab3.fki_id_transit;

CREATE INDEX fki_id_transit
    ON lab3.flight_schedule USING btree
    (id_transit ASC NULLS LAST)
    TABLESPACE pg_default;
```

Dashboard Properties SQL Statistics Dependencies Dependents postgres/daniil...

```sql
-- Table: lab3.passenger

-- DROP TABLE lab3.passenger;

CREATE TABLE IF NOT EXISTS lab3.passenger
(
    passport_number integer NOT NULL,
    first_name character varying(30) COLLATE pg_catalog."default" NOT NULL,
    last_name character varying(30) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT passenger_pkey PRIMARY KEY (passport_number)
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE lab3.passenger
    OWNER to daniil;
```

```sql
-- Table: lab3.plane

-- DROP TABLE lab3.plane;

CREATE TABLE IF NOT EXISTS lab3.plane
(
    id_plane integer NOT NULL,
    id_brand integer NOT NULL,
    release_date date NOT NULL,
    CONSTRAINT plane_pkey PRIMARY KEY (id_plane),
    CONSTRAINT plane_id_brand_fkey FOREIGN KEY (id_brand)
        REFERENCES lab3.plane_brand (id_brand) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE lab3.plane
    OWNER to daniil;
-- Index: fki_id_brand

-- DROP INDEX lab3.fki_id_brand;

CREATE INDEX fki_id_brand
    ON lab3.plane USING btree
    (id_brand ASC NULLS LAST)
    TABLESPACE pg_default;
```

```sql
-- Table: lab3.plane_brand

-- DROP TABLE lab3.plane_brand;

CREATE TABLE IF NOT EXISTS lab3.plane_brand
(
    speed integer NOT NULL,
    type character varying(30) COLLATE pg_catalog."default" NOT NULL,
    lifting_capacity integer NOT NULL,
    fuel_consumption integer NOT NULL,
    hours_flying integer NOT NULL,
    manufacturer character varying(30) COLLATE pg_catalog."default" NOT NULL,
    id_brand integer NOT NULL,
    seats_count integer NOT NULL,
    CONSTRAINT plane_brand_pkey PRIMARY KEY (id_brand),
    CONSTRAINT type_check CHECK (type::text = ANY (ARRAY['грузовой'::character varying::text, 'пассажирский'::character varying::text])) NOT VALID,
    CONSTRAINT man_check CHECK (manufacturer::text = ANY (ARRAY['Airbus'::character varying::text, 'Boeing'::character varying::text, 'ИЛ'::character varying::text, 'ТУ'::character varying::text])
    CONSTRAINT seats_check CHECK (seats_count > 0 AND seats_count < 100) NOT VALID
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE lab3.plane_brand
    OWNER to daniil;
```

Dashboard Properties SQL Statistics Dependencies Dependents postgres/daniil...

```sql
-- Table: lab3.purchase

-- DROP TABLE lab3.purchase;

CREATE TABLE IF NOT EXISTS lab3.purchase
(
    id_purchase integer NOT NULL,
    id_ticket integer NOT NULL,
    passport_number integer NOT NULL,
    seat_number integer NOT NULL,
    id_cashbox integer NOT NULL,
    CONSTRAINT purchase_pkey PRIMARY KEY (id_purchase),
    CONSTRAINT purchase_id_cashbox_fkey FOREIGN KEY (id_cashbox)
        REFERENCES lab3.cashbox (id_cashbox) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT purchase_id_ticket_fkey FOREIGN KEY (id_ticket)
        REFERENCES lab3.ticket (id_ticket) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT purchase_passport_number_fkey FOREIGN KEY (passport_number)
        REFERENCES lab3.passenger (passport_number) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE lab3.purchase
    OWNER to daniil;
-- Index: fki_id_cashbox

-- DROP INDEX lab3.fki_id_cashbox;

CREATE INDEX fki_id_cashbox
    ON lab3.purchase USING btree
    (id_cashbox ASC NULLS LAST)
    TABLESPACE pg_default;
-- Index: fki_id_pas

-- DROP INDEX lab3.fki_id_pas;

CREATE INDEX fki_id_pas
    ON lab3.purchase USING btree
    (passport_number ASC NULLS LAST)
```

```
-- Table: lab3.staff

-- DROP TABLE lab3.staff;

CREATE TABLE IF NOT EXISTS lab3.staff
(
    status character varying(30) COLLATE pg_catalog."default" NOT NULL,
    emp_passport_number integer NOT NULL,
    id_staff integer NOT NULL,
    CONSTRAINT staff_pkey PRIMARY KEY (id_staff),
    CONSTRAINT id_emp FOREIGN KEY (emp_passport_number)
        REFERENCES lab3.employer (passport_number) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT stauts_check CHECK (status::text = ANY (ARRAY['допущен'::character varying::text, 'не допущен'::character varying::text])) NOT VALID
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE lab3.staff
    OWNER to daniil;
-- Index: fki_emp_passport_number

-- DROP INDEX lab3.fki_emp_passport_number;

CREATE INDEX fki_emp_passport_number
    ON lab3.staff USING btree
    (emp_passport_number ASC NULLS LAST)
    TABLESPACE pg_default;
```

```
-- Table: lab3.ticket

-- DROP TABLE lab3.ticket;

CREATE TABLE IF NOT EXISTS lab3.ticket
(
    id_ticket integer NOT NULL,
    id_flight integer NOT NULL,
    price integer NOT NULL,
    CONSTRAINT ticket_pkey PRIMARY KEY (id_ticket),
    CONSTRAINT id_flight FOREIGN KEY (id_flight)
        REFERENCES lab3.flight (id_flight) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE lab3.ticket
    OWNER to daniil;
-- Index: fki_id_flight

-- DROP INDEX lab3.fki_id_flight;

CREATE INDEX fki_id_flight
    ON lab3.ticket USING btree
    (id_flight ASC NULLS LAST)
    TABLESPACE pg_default;
```

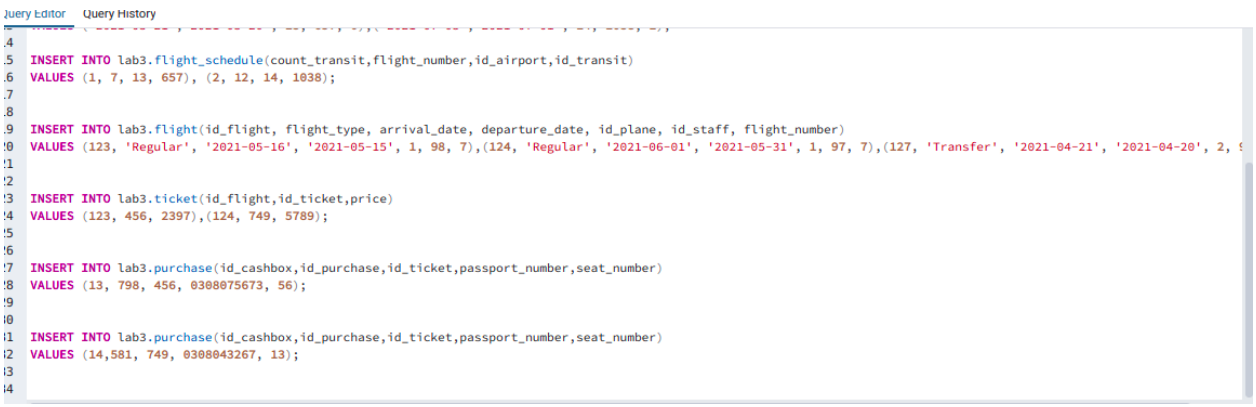
Dashboard Properties SQL Statistics Dependencies Dependents postgres/daniil...

```sql
-- Table: lab3.transit

-- DROP TABLE lab3.transit;

CREATE TABLE IF NOT EXISTS lab3.transit
(
    id_transit integer NOT NULL,
    transit_order integer NOT NULL,
    id_airport integer NOT NULL,
    arrival_date date NOT NULL,
    departure_date date NOT NULL,
    CONSTRAINT transit_pkey PRIMARY KEY (id_transit),
    CONSTRAINT id_airport FOREIGN KEY (id_airport)
        REFERENCES lab3.airports (id_airport) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT dep_date_check CHECK (departure_date > '1900-01-01'::date) NOT VALID,
    CONSTRAINT ar_date_check CHECK (arrival_date > departure_date) NOT VALID
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE lab3.transit
    OWNER to daniil;
```

6. Заполнение данными

Query Editor Query History

```sql
INSERT INTO lab3.employer(first_name,last_name, job, passport_number)
VALUES ('Артем', 'Галкин','первый пилот', 0218321673),('Наталья', 'Леонова','стюардесса', 0243233233),('Олег', 'Ковалев','второй пилот', 0756409567);


INSERT INTO lab3.staff(status, emp_passport_number, id_staff)
VALUES ('допущен',0218321673, 98),('не допущен',0243233233, 97),('допущен',0756409567, 96);

INSERT INTO lab3.plane(id_brand,id_plane,release_date)
VALUES (103, 1, '2016-03-21'),(102, 2, '2013-07-05');


INSERT INTO lab3.transit(arrival_date,departure_date,id_airport,id_transit,transit_order)
VALUES ('2021-03-21','2021-03-20', 13, 657, 0),('2021-07-03','2021-07-01', 14, 1038, 1);

INSERT INTO lab3.flight_schedule(count_transit,flight_number,id_airport,id_transit)
VALUES (1, 7, 13, 657), (2, 12, 14, 1038);


INSERT INTO lab3.flight(id_flight, flight_type, arrival_date, departure_date, id_plane, id_staff, flight_number)
VALUES (123, 'Regular', '2021-05-16', '2021-05-15', 1, 98, 7),(124, 'Regular', '2021-06-01', '2021-05-31', 1, 97, 7),(127, 'Transfer', '2021-04-21', '2021-04-20', 2,
```



ВЫВОДЫ

Программа pgAdmin позволяет создавать базы данных на высоком уровне: либо напрямую посредством взаимодействия с ее GUI, или же через работу на встроенном генераторе ER диаграмм. Однако, каким способом диаграмма не была создана, всегда сохраняется возможность увидеть какими SQL командами была создана диаграмма. Видеть код, создаваемый нажатием кнопки в интерфейсе, оказалось полезным для нахождения ошибок и общего понимания работы программы.