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 таблиц рабочими данными
по дисциплине: Проектирование и реализация баз данных

Специальность:
45.03.04 Интеллектуальные системы в гуманитарной сфере

Проверил:
Говорова М.М.
Дата: «14» марта 2022 г.
Оценка _____

Выполнила:
студентка группы К3243
Нургазизова А.Р.

Санкт-Петербург 2021/2022

Цель работы

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

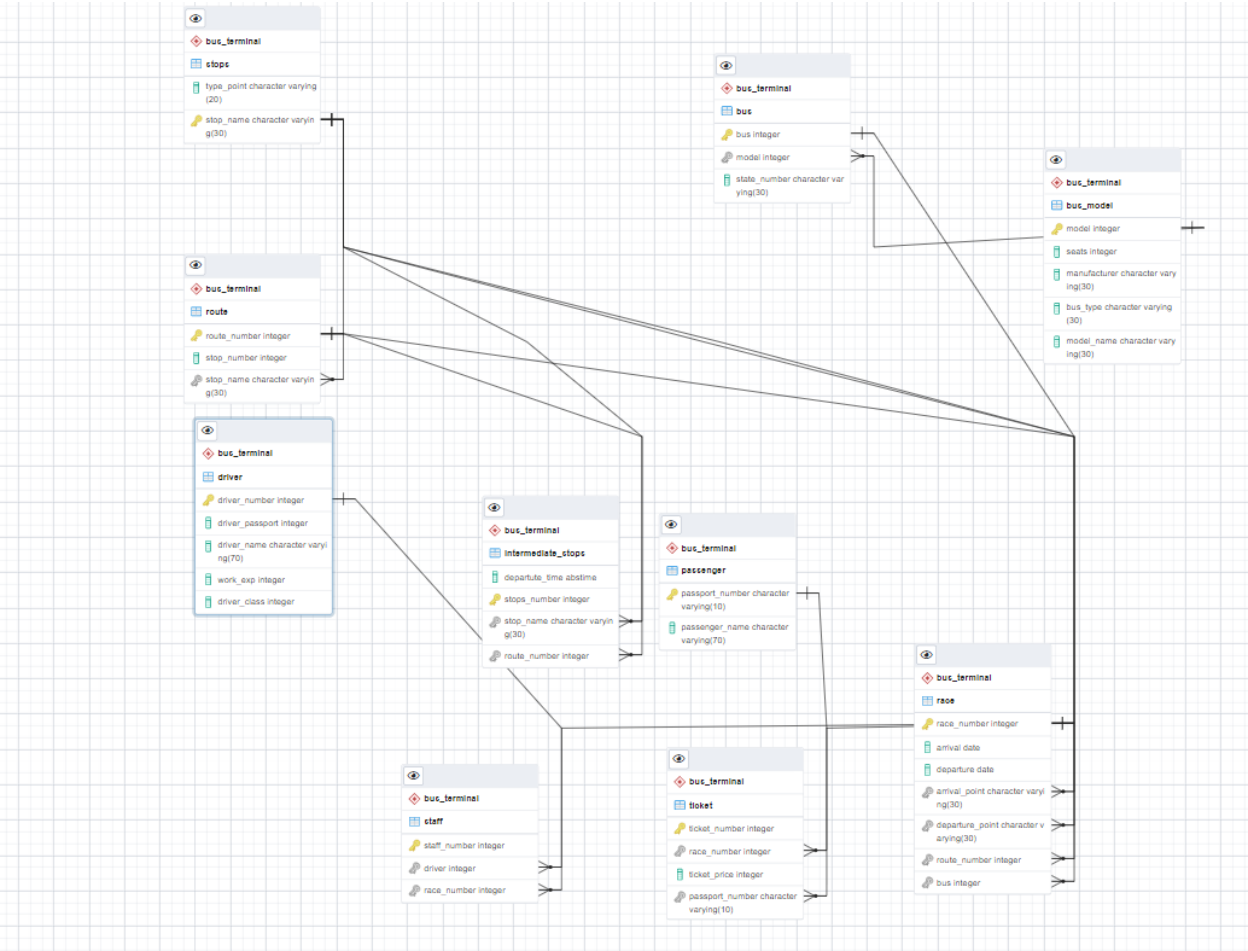
Практическое задание

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.
6. Создать резервную копию БД.
7. Восстановить БД.

Выполнение

I. Наименование БД

bus_terminal.



II. Схема логической модели БД, сгенерированная в ERD

Рис. 1 — Схема логической модели БД, сгенерированная в ERD

III. Dump, содержащий скрипты работы с БД

1. Создание БД

```
CREATE SCHEMA IF NOT EXISTS bus_terminal
    AUTHORIZATION postgres; // создание БД
```

2. Создание таблиц

```
CREATE TABLE IF NOT EXISTS bus_terminal.bus
(
    bus integer NOT NULL,
    model integer NOT NULL,
    state_number character varying(30) COLLATE pg_catalog."default",
    CONSTRAINT bus_keyp PRIMARY KEY (bus),
    CONSTRAINT bus_fkey FOREIGN KEY (model)
```

```
        REFERENCES bus_terminal.bus_model (model) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
    )
    WITH (
      OIDS = FALSE
    )
    TABLESPACE pg_default;

    ALTER TABLE IF EXISTS bus_terminal.bus
      OWNER to postgres;
    -- Index: fki_bus_fkey

    -- DROP INDEX IF EXISTS bus_terminal.fki_bus_fkey;

    CREATE INDEX IF NOT EXISTS fki_bus_fkey
      ON bus_terminal.bus USING btree
      (model ASC NULLS LAST)
TABLESPACE pg_default;// создание таблицы bus

    CREATE TABLE IF NOT EXISTS bus_terminal.bus_model
    (
      model integer NOT NULL,
      seats integer NOT NULL,
      manufacturer character varying(30) COLLATE pg_catalog."default" NOT
    NULL,
      bus_type character varying(30) COLLATE pg_catalog."default" NOT
    NULL,
      model_name character varying(30) COLLATE pg_catalog."default" NOT
    NULL,
```

```
    CONSTRAINT bus_model_keyp PRIMARY KEY (model),
    CONSTRAINT seats CHECK (seats >= 10),
    CONSTRAINT seats_end CHECK (seats <= 60)
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS bus_terminal.bus_model
OWNER to postgres; // создание таблицы bus_model (ограничения Check)

CREATE TABLE IF NOT EXISTS bus_terminal.driver
(
    driver_number integer NOT NULL,
    driver_passport integer NOT NULL,
    driver_name character varying(70) COLLATE pg_catalog."default" NOT
NULL,
    work_exp integer NOT NULL,
    driver_class integer,
    CONSTRAINT driver_keyp PRIMARY KEY (driver_number),
    CONSTRAINT driver_passport UNIQUE (driver_passport)
        INCLUDE(driver_passport),
    CONSTRAINT work_ex CHECK (work_exp > 0)
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;
```

```
    ALTER TABLE IF EXISTS bus_terminal.driver
OWNER to postgres;// создание таблицы driver (ограничения Check, Unique)

    CREATE TABLE IF NOT EXISTS bus_terminal.intermediate_stops
    (
        departute_time abstime NOT NULL,
        stops_number integer NOT NULL,
        stop_name character varying(30) COLLATE pg_catalog."default",
        route_number integer,
        CONSTRAINT stops_keypr PRIMARY KEY (stops_number)
            INCLUDE(stops_number),
        CONSTRAINT route_frkey FOREIGN KEY (route_number)
            REFERENCES bus_terminal.route (route_number) MATCH SIMPLE
            ON UPDATE NO ACTION
            ON DELETE NO ACTION
            NOT VALID,
        CONSTRAINT stops_frkey FOREIGN KEY (stop_name)
            REFERENCES bus_terminal.stops (stop_name) MATCH SIMPLE
            ON UPDATE NO ACTION
            ON DELETE NO ACTION
    )
    WITH (
        OIDS = FALSE
    )
    TABLESPACE pg_default;

    ALTER TABLE IF EXISTS bus_terminal.intermediate_stops
        OWNER to postgres;
    -- Index: fki_stops_frkey
```

```
-- DROP INDEX IF EXISTS bus_terminal.fki_stops_frkey;

CREATE INDEX IF NOT EXISTS fki_stops_frkey
    ON bus_terminal.intermediate_stops USING btree
    (stop_name COLLATE pg_catalog."default" ASC NULLS LAST)
    TABLESPACE pg_default; // создание таблицы intermediate_stops

CREATE TABLE IF NOT EXISTS bus_terminal.passenger
(
    passport_number character varying(10) COLLATE pg_catalog."default" NOT NULL,
    passenger_name character varying(70) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT passenger_keyp PRIMARY KEY (passport_number)
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS bus_terminal.passenger
OWNER to postgres;// создание таблицы passenger

CREATE TABLE IF NOT EXISTS bus_terminal.race
(
    race_number integer NOT NULL,
    arrival date NOT NULL,
    departure date NOT NULL,
    arrival_point character varying(30) COLLATE pg_catalog."default" NOT NULL,
```

```
    departure_point character varying(30) COLLATE pg_catalog."default"
NOT NULL,
    route_number integer NOT NULL,
    bus integer,
    CONSTRAINT race_keyp PRIMARY KEY (race_number),
    CONSTRAINT arr_frkey FOREIGN KEY (arrival_point)
        REFERENCES bus_terminal.stops (stop_name) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT bus_frkey FOREIGN KEY (bus)
        REFERENCES bus_terminal.bus (bus) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT dep_frkey FOREIGN KEY (departure_point)
        REFERENCES bus_terminal.stops (stop_name) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT race_fkey FOREIGN KEY (route_number)
        REFERENCES bus_terminal.route (route_number) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT race CHECK (arrival >= departure)
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;
```

```
    ALTER TABLE IF EXISTS bus_terminal.race
        OWNER to postgres;
    -- Index: fki_race_fkey

    -- DROP INDEX IF EXISTS bus_terminal.fki_race_fkey;

    CREATE INDEX IF NOT EXISTS fki_race_fkey
        ON bus_terminal.race USING btree
        (route_number ASC NULLS LAST)
TABLESPACE pg_default;// создание таблицы race

    CREATE TABLE IF NOT EXISTS bus_terminal.route
    (
        route_number integer NOT NULL,
        stop_number integer NOT NULL,
        stop_name character varying(30) COLLATE pg_catalog."default",
        CONSTRAINT route_keyp PRIMARY KEY (route_number),
        CONSTRAINT stops_fkey FOREIGN KEY (stop_name)
            REFERENCES bus_terminal.stops (stop_name) MATCH SIMPLE
            ON UPDATE NO ACTION
            ON DELETE NO ACTION
    )
    WITH (
        OIDS = FALSE
    )
    TABLESPACE pg_default;

    ALTER TABLE IF EXISTS bus_terminal.route
        OWNER to postgres;
```

```
    -- Index: fki_stops_fkey

    -- DROP INDEX IF EXISTS bus_terminal.fki_stops_fkey;

    CREATE INDEX IF NOT EXISTS fki_stops_fkey
        ON bus_terminal.route USING btree
        (stop_name COLLATE pg_catalog."default" ASC NULLS LAST)
TABLESPACE pg_default;// создание таблицы route

    CREATE TABLE IF NOT EXISTS bus_terminal.staff
    (
        staff_number integer NOT NULL,
        driver integer NOT NULL,
        race_number integer NOT NULL,
        CONSTRAINT staff_keyp PRIMARY KEY (staff_number),
        CONSTRAINT driver_fkey FOREIGN KEY (driver)
            REFERENCES bus_terminal.driver (driver_number) MATCH SIMPLE
            ON UPDATE NO ACTION
            ON DELETE NO ACTION,
        CONSTRAINT staff_fkey FOREIGN KEY (race_number)
            REFERENCES bus_terminal.race (race_number) MATCH SIMPLE
            ON UPDATE NO ACTION
            ON DELETE NO ACTION
    )
    WITH (
        OIDS = FALSE
    )
    TABLESPACE pg_default;

    ALTER TABLE IF EXISTS bus_terminal.staff
```

```
    OWNER to postgres;
-- Index: fki_driver_fkey

-- DROP INDEX IF EXISTS bus_terminal.fki_driver_fkey;

CREATE INDEX IF NOT EXISTS fki_driver_fkey
    ON bus_terminal.staff USING btree
    (driver ASC NULLS LAST)
    TABLESPACE pg_default;
-- Index: fki_staff_fkey

-- DROP INDEX IF EXISTS bus_terminal.fki_staff_fkey;

CREATE INDEX IF NOT EXISTS fki_staff_fkey
    ON bus_terminal.staff USING btree
    (race_number ASC NULLS LAST)
TABLESPACE pg_default; // создание таблицы staff

CREATE TABLE IF NOT EXISTS bus_terminal.stops
(
    type_point character varying(20) COLLATE pg_catalog."default" NOT
NULL,
    stop_name character varying(30) COLLATE pg_catalog."default" NOT
NULL,
    CONSTRAINT stops_keyp PRIMARY KEY (stop_name)
        INCLUDE(stop_name)
)
WITH (
    OIDS = FALSE
)
```

```
    TABLESPACE pg_default;

    ALTER TABLE IF EXISTS bus_terminal.stops
OWNER to postgres; // создание таблицы stops

    CREATE TABLE IF NOT EXISTS bus_terminal.ticket
    (
        ticket_number integer NOT NULL,
        race_number integer NOT NULL,
        ticket_price integer NOT NULL,
        passport_number character varying(10) COLLATE pg_catalog."default"
    NOT NULL,
        CONSTRAINT ticket_keyp PRIMARY KEY (ticket_number),
        CONSTRAINT pass_fkey FOREIGN KEY (passport_number)
            REFERENCES bus_terminal.passenger (passport_number) MATCH
    SIMPLE
            ON UPDATE NO ACTION
            ON DELETE NO ACTION,
        CONSTRAINT ticket_fkey FOREIGN KEY (race_number)
            REFERENCES bus_terminal.race (race_number) MATCH SIMPLE
            ON UPDATE NO ACTION
            ON DELETE NO ACTION,
        CONSTRAINT ticket_pr_max CHECK (ticket_price <= 5000),
        CONSTRAINT ticket_pr CHECK (ticket_price > 0)
    )
    WITH (
        OIDS = FALSE
    )
    TABLESPACE pg_default;
```

```
ALTER TABLE IF EXISTS bus_terminal.ticket
    OWNER to postgres;
-- Index: fki_pass_fkey

-- DROP INDEX IF EXISTS bus_terminal.fki_pass_fkey;

CREATE INDEX IF NOT EXISTS fki_pass_fkey
    ON bus_terminal.ticket USING btree
    (passport_number COLLATE pg_catalog."default" ASC NULLS LAST)
    TABLESPACE pg_default;
-- Index: fki_ticket_fkey

-- DROP INDEX IF EXISTS bus_terminal.fki_ticket_fkey;

CREATE INDEX IF NOT EXISTS fki_ticket_fkey
    ON bus_terminal.ticket USING btree
    (race_number ASC NULLS LAST)
  TABLESPACE pg_default; // создание таблицы ticket(ограничение Check)
```

3. Вставка данных

```
INSERT INTO bus_terminal.bus_model
VALUES (1, 30, ‘LADA’, ‘средний’, ‘Автобус’)

INSERT INTO bus_terminal.bus_model
VALUES (2, 10, ‘LADA’, ‘малый’, ‘Автобус_малый’)

INSERT INTO bus_terminal.bus_model
VALUES (3, 60, ‘LADA’, ‘большой’, ‘Автобус_большой’) // вставка значений в таблицу bus_model
```

INSERT INTO bus_terminal.bus
VALUES (1, 1, 1)

INSERT INTO bus_terminal.bus
VALUES (2, 2, 2)

INSERT INTO bus_terminal.bus
VALUES (3, 3, 3)
// *вставка значений в таблицу bus*

INSERT INTO bus_terminal.passenger
VALUES (‘6716529183’, ‘Железнова Мария’)

INSERT INTO bus_terminal.passenger
VALUES (‘6715432678’, ‘Мин Юнги’)

INSERT INTO bus_terminal.passenger
VALUES (‘2040532681’, ‘Зубова Екатерина’)

INSERT INTO bus_terminal.passenger
VALUES (‘4387290146’, ‘Нургазизова Айзиля’) // *вставка значений в таблицу passenger*

INSERT INTO bus_terminal.route
VALUES (1, 1, ‘Екатеринбург’)

INSERT INTO bus_terminal.route
VALUES (2, 2, ‘Казань’)

INSERT INTO bus_terminal.route

VALUES (3, 3, ‘Челябинск’) // *вставка значений в таблицу route*

INSERT INTO bus_terminal.race
VALUES (1, ‘20210325’, ‘20210320’, ‘Екатеринбург’, ‘Казань’, 1)

INSERT INTO bus_terminal.race
VALUES (2, ‘20211225’, ‘20211214’, ‘Казань’, ‘Челябинск’, 2)

INSERT INTO bus_terminal.race
VALUES (3, ‘20210506’, ‘20210503’, ‘Челябинск’, ‘Екатеринбург’, 2) // *вставка значений в таблицу race*

INSERT INTO bus_terminal.staff
VALUES (1, 1, 1)

INSERT INTO bus_terminal.staff
VALUES (2, 2, 2)

INSERT INTO bus_terminal.staff
VALUES (3, 3, 3) // *вставка значений в таблицу staff*

INSERT INTO bus_terminal.stops
VALUES (‘город’, ‘Екатеринбург’)

INSERT INTO bus_terminal.stops
VALUES (‘город’, ‘Казань’)

INSERT INTO bus_terminal.stops
VALUES (‘город’, ‘Челябинск’) // *вставка значений в таблицу stops*

INSERT INTO bus_terminal.ticket
VALUES (1, 1, 500, ‘6716529183’, ‘Казань’, ‘Екатеринбург’)

INSERT INTO bus_terminal.ticket
VALUES (2, 2, 500, ‘6715432678’, ‘Казань’, ‘Екатеринбург’)

INSERT INTO bus_terminal.ticket
VALUES (3, 3, 500, ‘2040532681’, ‘Казань’, ‘Челябинск’)

INSERT INTO bus_terminal.ticket
VALUES (4, 3, 500, ‘4387290146’, ‘Екатеринбург’, ‘Челябинск’) // *вставка значений в таблицу ticket*

INSERT INTO bus_terminal.driver
VALUES (1, 5677, ‘Нургазизова Айгуль’, 5)

INSERT INTO bus_terminal.driver
VALUES (2, 5718, ‘Цуриков Даниил’, 3)

INSERT INTO bus_terminal.driver
VALUES (3, 6743, ‘Гусенова Аида’, 10) // *вставка значений в таблицу driver*

Выводы

В ходе выполнения лабораторной работы была создана база данных, в неё были добавлены значения, а также созданы первичные и внешние ключи, ограничения для полей. Были изучены основы pgAdmin 4.