Университет ИТМО

Лабораторная работа № 3.2

«Создание таблиц базы данных PostgreSQL. Заполнение таблиц рабочими данными»

Выполнила: Анисимова Ксения Сергеевна
Группа: К3241
Преподаватель: Говорова Марина Михайловна

Санкт – Петербург
2023

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.
7. Восстановить БД.

Ход работы:

1. Наименование БД

a) Program (*program_id* Идентификатор программы, start_date дата начала, *end_date* дата конца, *program_name* название программы, *program_type_id* идентификатор типа программы)

b) Program type (*program_type_id* Идентификатор типа программы, *program_type_name* название, *qualification* квалификация, *document_type* тип документа)

c) Discipline (*discipline_id* Идентификатор дисциплины, *discipline_name* название дисциплины, *lec_hours* лекционные часы, *lab_hours* лабораторные часы, *prac_hours* практические часы)

d) Group (*group_id* Идентификатор группы, *program_id* идентификатор программы, *group_number* номер группы, *volume* вместимость)

e) Listener (*passport* Номер паспорта, *phone_number* контакты, *listener_name* имя слушателя, *listener_surname* фамилия слушателя)

f) Education (*passport* Номер паспорта, *group_id* идентификатор группы, *status* статус, *diploma_number* номер документа об обучении)

g) Class (*class_id* Идентификатор занятия, *room_id* идентификатор аудитории, *discipline_id* идентификатор дисциплины, *group id* идентификатор группы, *teacher number* табельный номер, *date* дата занятия, *class_status* статус занятия, *class_num* номер пары, *class_type* тип занятия)

h) Classroom (*room_id* Идентификатор аудитории, *address* адрес, *type* тип аудитории, *room_number* номер аудитории)

i) Преподаватель (*teacher_number* Табельный номер, *teacher_name* имя, *teacher_surname* фамилия, *teacher_middlename* отчество, *occupation_id* идентификатор должности)

j) Должность (*occupation_id* Идентификатор должности, *occupation_name* название)

2. Схема логической модели базы данных, сгенерированная в Generate ERD

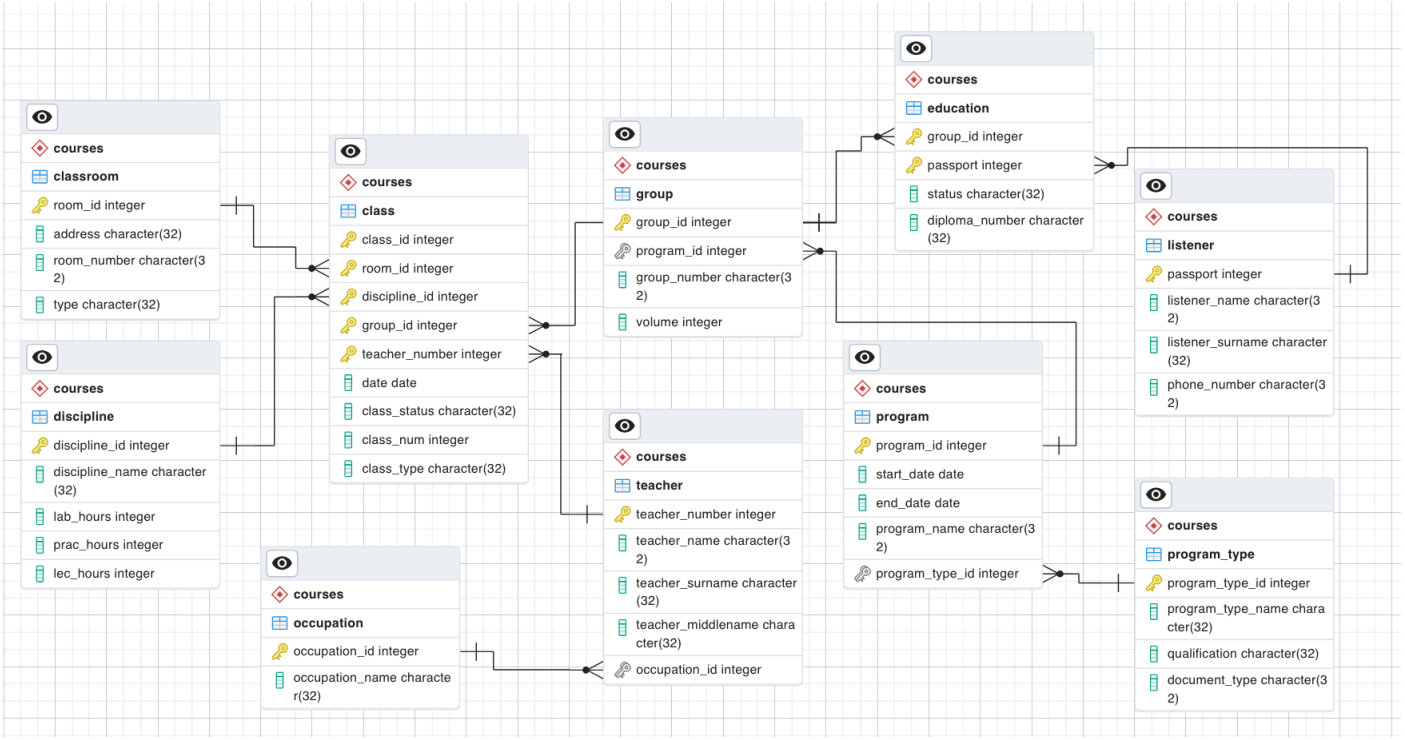

Рисунок 1

3. Dump, скомпилированный pg_Backup, содержащий скрипты работы с БД:

```
CREATE SCHEMA courses;
ALTER SCHEMA courses OWNER TO postgres;
SET default_tablespace = '';
SET default_table_access_method = heap;

CREATE TABLE courses.class (
    class_id integer NOT NULL,
    room_id integer NOT NULL,
    discipline_id integer NOT NULL,
    group_id integer NOT NULL,
    teacher_number integer NOT NULL,
    date date NOT NULL,
    class_status character(32) NOT NULL,
    class_num integer NOT NULL,
    class_type character(32) NOT NULL,
    CONSTRAINT class_class_type_check CHECK (((class_type = 'Лабораторная'::bpchar) OR (class_type =
'Практика'::bpchar) OR (class_type = 'Лекция'::bpchar)))
);
ALTER TABLE courses.class OWNER TO postgres;

CREATE TABLE courses.classroom (
    room_id integer NOT NULL,
    address character(32) NOT NULL,
    room_number character(32) NOT NULL,
    type character(32) NOT NULL
);
ALTER TABLE courses.classroom OWNER TO postgres;

CREATE TABLE courses.discipline (
```

```
    discipline_id integer NOT NULL,
    discipline_name character(32) NOT NULL,
    lab_hours integer NOT NULL,
    prac_hours integer NOT NULL,
    lec_hours integer NOT NULL,
    CONSTRAINT lab_hours_chk CHECK (((lab_hours > 0) AND (lab_hours < 300))),
    CONSTRAINT lec_hours_chk CHECK (((lec_hours > 0) AND (lec_hours < 300))),
    CONSTRAINT prac_hours_chk CHECK (((prac_hours > 0) AND (prac_hours < 300)))
);
ALTER TABLE courses.discipline OWNER TO postgres;

CREATE TABLE courses.education (
    group_id integer NOT NULL,
    passport integer NOT NULL,
    status character(32) NOT NULL,
    diploma_number character(32) NOT NULL
);
ALTER TABLE courses.education OWNER TO postgres;

CREATE TABLE courses.group (
    group_id integer NOT NULL,
    program_id integer NOT NULL,
    group_number character(32) NOT NULL,
    volume integer NOT NULL
);
ALTER TABLE courses."group" OWNER TO postgres;

CREATE TABLE courses.listener (
    passport integer NOT NULL,
    listener_name character(32) NOT NULL,
    listener_surname character(32) NOT NULL,
    phone_number character(32) NOT NULL
);
ALTER TABLE courses.listener OWNER TO postgres;

CREATE TABLE courses.occupation (
    occupation_id integer NOT NULL,
    occupation_name character(32) NOT NULL
);
ALTER TABLE courses.occupation OWNER TO postgres;

CREATE TABLE courses.program (
    program_id integer NOT NULL,
    start_date date NOT NULL,
    end_date date NOT NULL,
    program_name character(32) NOT NULL,
    program_type_id integer NOT NULL
);
ALTER TABLE courses.program OWNER TO postgres;

CREATE TABLE courses.program_type (
    program_type_id integer NOT NULL,
    program_type_name character(32) NOT NULL,
    qualification character(32) NOT NULL,
    document_type character(32) NOT NULL
);
ALTER TABLE courses.program_type OWNER TO postgres;

CREATE TABLE courses.teacher (
    teacher_number integer NOT NULL,
    teacher_name character(32) NOT NULL,
    teacher_surname character(32) NOT NULL,
    teacher_middlename character(32) NOT NULL,
    occupation_id integer NOT NULL
);
ALTER TABLE courses.teacher OWNER TO postgres;
```

```
COPY courses.class (class_id, room_id, discipline_id, group_id, teacher_number, date, class_status,
class_num, class_type) FROM stdin;
\.
COPY courses.classroom (room_id, address, room_number, type) FROM stdin;
\.
COPY courses.discipline (discipline_id, discipline_name, lab_hours, prac_hours, lec_hours) FROM stdin;
\.
COPY courses.education (group_id, passport, status, diploma_number) FROM stdin;
\.
COPY courses.group (group_id, program_id, group_number, volume) FROM stdin;
\.
COPY courses.listener (passport, listener_name, listener_surname, phone_number) FROM stdin;
\.
COPY courses.occupation (occupation_id, occupation_name) FROM stdin;
\.
COPY courses.program (program_id, start_date, end_date, program_name, program_type_id) FROM stdin;
\.
COPY courses.program_type (program_type_id, program_type_name, qualification, document_type) FROM stdin;
\.
COPY courses.teacher (teacher_number, teacher_name, teacher_surname, teacher_middlename, occupation_id)
FROM stdin;
\.

ALTER TABLE ONLY courses.class
    ADD CONSTRAINT class_pkey PRIMARY KEY (class_id, room_id, discipline_id, group_id, teacher_number);
ALTER TABLE courses.classroom
    ADD CONSTRAINT classroom_room_id_check CHECK ((room_id > 0)) NOT VALID;
ALTER TABLE courses.discipline
    ADD CONSTRAINT discipline_id_chk CHECK ((discipline_id > 0)) NOT VALID;
ALTER TABLE ONLY courses.discipline
    ADD CONSTRAINT discipline_pkey PRIMARY KEY (discipline_id);
ALTER TABLE ONLY courses.education
    ADD CONSTRAINT education_pkey PRIMARY KEY (group_id, passport);
ALTER TABLE courses."group"
    ADD CONSTRAINT group_group_id_check CHECK ((group_id > 0)) NOT VALID;
ALTER TABLE ONLY courses."group"
    ADD CONSTRAINT group_pkey PRIMARY KEY (group_id);
ALTER TABLE courses."group"
    ADD CONSTRAINT group_volume_check CHECK (((volume > 0) AND (volume < 1000))) NOT VALID;
ALTER TABLE courses.listener
    ADD CONSTRAINT listener_phone_number_check CHECK ((phone_number ~~ '8[0-9]{10}'::text)) NOT VALID;
ALTER TABLE ONLY courses.listener
    ADD CONSTRAINT listener_pkey PRIMARY KEY (passport);
ALTER TABLE courses.occupation
    ADD CONSTRAINT occupation_occupation_id_check CHECK ((occupation_id > 0)) NOT VALID;
ALTER TABLE ONLY courses.occupation
    ADD CONSTRAINT occupation_pkey PRIMARY KEY (occupation_id)
ALTER TABLE ONLY courses.program
    ADD CONSTRAINT program_pkey PRIMARY KEY (program_id);
ALTER TABLE courses.program
    ADD CONSTRAINT program_program_id_check CHECK ((program_id > 0)) NOT VALID;
ALTER TABLE ONLY courses.program_type
    ADD CONSTRAINT program_type_pkey PRIMARY KEY (program_type_id);
ALTER TABLE ONLY courses.classroom
    ADD CONSTRAINT room_pkey PRIMARY KEY (room_id);
ALTER TABLE ONLY courses.teacher
    ADD CONSTRAINT teacher_pkey PRIMARY KEY (teacher_number);
ALTER TABLE courses.teacher
    ADD CONSTRAINT teacher_teacher_number_check CHECK ((teacher_number > 0)) NOT VALID;
ALTER TABLE ONLY courses.class
    ADD CONSTRAINT class1_discipline_id_fkey FOREIGN KEY (discipline_id) REFERENCES
courses.discipline(discipline_id);
ALTER TABLE ONLY courses.class
    ADD CONSTRAINT class1_group_id_fkey FOREIGN KEY (group_id) REFERENCES courses."group"(group_id);
ALTER TABLE ONLY courses.class
    ADD CONSTRAINT class1_room_id_fkey FOREIGN KEY (room_id) REFERENCES courses.classroom(room_id);
ALTER TABLE ONLY courses.class
```

```
    ADD CONSTRAINT class1_teacher_number_fkey FOREIGN KEY (teacher_number) REFERENCES
courses.teacher(teacher_number);
ALTER TABLE ONLY courses.education
    ADD CONSTRAINT education_group_id_fkey FOREIGN KEY (group_id) REFERENCES courses."group"(group_id);--
ALTER TABLE ONLY courses.education
    ADD CONSTRAINT education_passport_fkey FOREIGN KEY (passport) REFERENCES courses.listener(passport);
ALTER TABLE ONLY courses."group"
    ADD CONSTRAINT group_program_id_fkey FOREIGN KEY (program_id) REFERENCES courses.program(program_id);
ALTER TABLE ONLY courses.program
    ADD CONSTRAINT program_program_type_id_fkey FOREIGN KEY (program_type_id) REFERENCES
courses.program_type(program_type_id) NOT VALID;
ALTER TABLE ONLY courses.teacher
    ADD CONSTRAINT teacher_occupation_id_fkey FOREIGN KEY (occupation_id) REFERENCES
courses.occupation(occupation_id);
```

Вывод:

В ходе выполнения лабораторной работы была создана база данных с использованием pgAdmin 4. Внутри БД были созданы схема, таблицы. Были заданы ограничения Check, Foreign Key. С помощью Query Tool таблицы были заполнены данными, а с помощью утилит pg_Dump и pg_Restore у БД была создана резервная копия и проведено восстановление данных.