Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики

Лабораторная работа№ 3

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

Выполнил: Евдокимов Владислав Борисович
Группа: К3242
Преподаватель: Говорова Марина Михайловна

Санкт-Петербург
2021

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.

ТЕХНОЛОГИЯ ВЫПОЛНЕНИЯ РАБОТЫ:

1. Название БД

Вариант 7. «Курсы»

Описание предметной области: Подразделение занимается организацией внебюджетного образования. Имеется несколько типов краткосрочных курсов, предназначенных для определенных специальностей, связанных с программным обеспечением ИТ. Каждый тип курсов имеет определенную длительность и свой перечень изучаемых дисциплин. На каждую программу может быть набрано несколько групп обучающихся. По каждой дисциплине могут проводиться лекционные и лабораторные занятия. Подразделение обеспечивает следующие ресурсы: учебные классы, лекционные аудитории и преподавателей. Необходимо составить расписание занятий.

БД должна содержать следующий минимальный набор сведений: Фамилия слушателя. Имя слушателя. Паспортные данные. Контакты. Код программы. Программа. Тип программы. Объем часов. Номер группы. максимальное количество человек в группе (для набора). Дата начала обучения. Дата окончания обучения. Название дисциплины. Количество часов. Дата занятий. Номер пары. Номер аудитории. Тип аудитории. Адрес площадки. Вид занятий (лекционные, практические или лабораторные). Фамилия преподавателя. Имя и отчество преподавателя. Должность преподавателя. Дисциплины, которые может вести преподаватель.

Состав реквизитов сущностей:

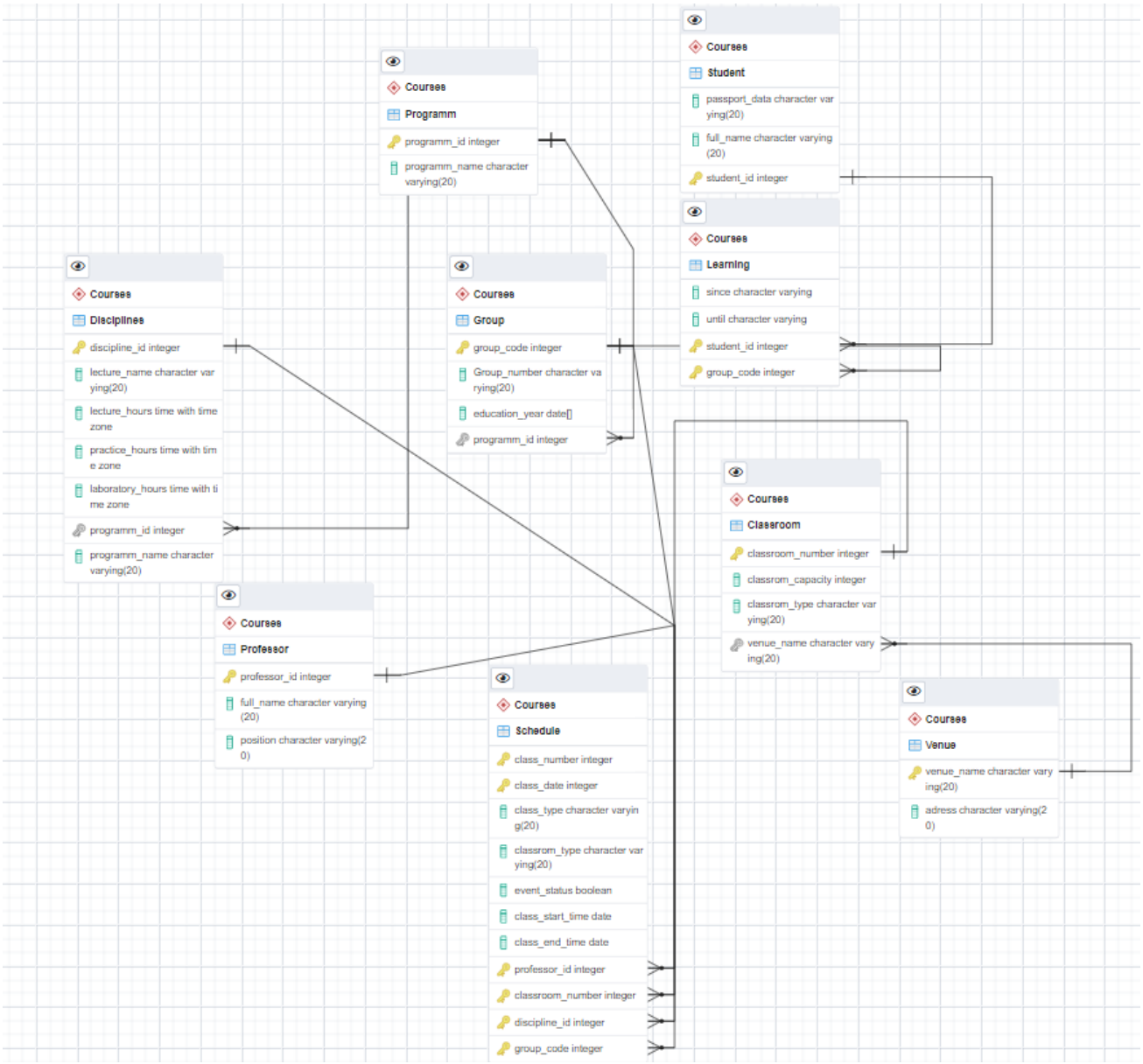
a) **Направление (**Код программы, наименование)
b) **Дисциплины** (ID дисциплины, код направления, название дисциплины, лекционные часы, лабораторные часы, практические часы)
c) **Группа** (Код группы, номер группы, год обучения, код направления)
d) **Слушатель** (ID слушателя, контакты, имя, фамилия, код группы, паспортные данные)
e) **Расписание** (Код расписания, ID преподавателя, ID дисциплины, код группы, тип занятий, номер пары, кол-во часов, номер аудитории, статус проведения, даты занятий, тип аудитории, номер класса, врем конца занятий, время начала занятий)
f) **Площадка проведения** (Название, адрес)

g) **Аудитория** (Номер аудитории, тип аудитории, вместимость, название площадки)

h) **Преподаватель** (ID преподавателя, ФИО, должность)

i)

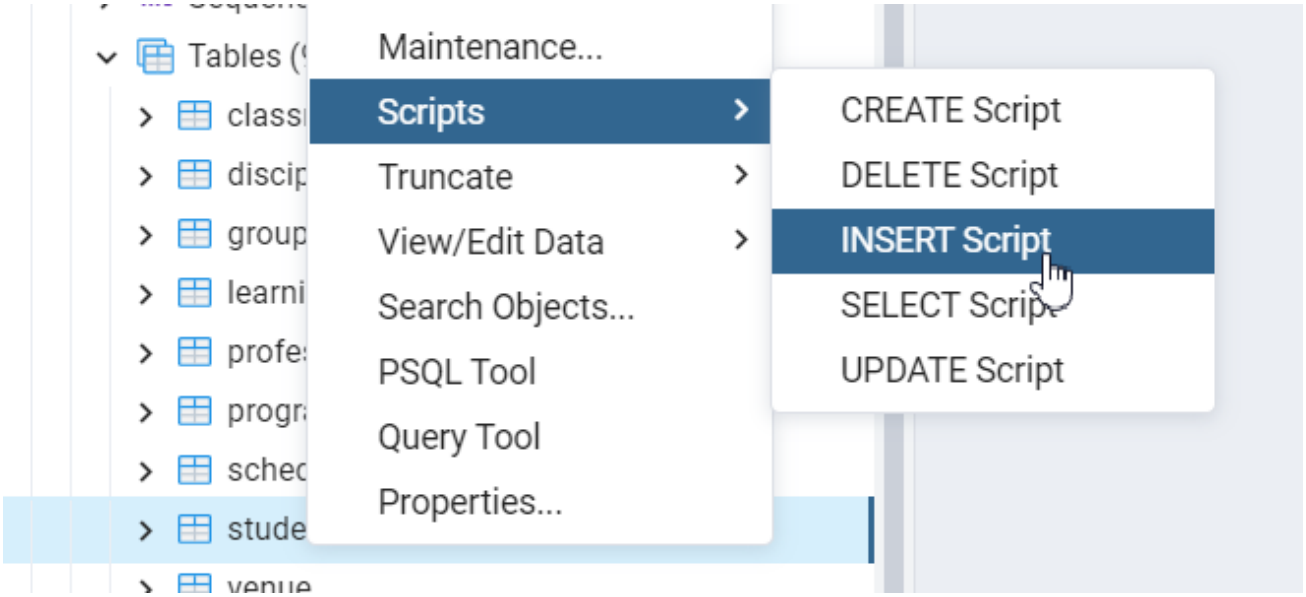
2. Схема логической модели базы данных:



Серьезным ограничением или даже багом программы является невозможность задать ограничение для столбца, если его имя содержит символы верхнего регистра. Причем программа никак не обозначает этот факт.

Заполнение таблиц рабочими данными

Чтобы избежать конфликтов при внесении данных, желательно использовать скрипты.



Query Editor | Query History

```
1  INSERT INTO courses.student(
2      "passport_data ", "full_name ", "student_id ")
3      VALUES ('05 15 320322', 'Иванов Иван Иванович', '347594');
```

Data Output | Explain | Messages | Notifications

	passport_data character varying (40)	full_name character varying (40)	student_id [PK] integer
1	05 15 320322	Иванов Иван Иванович	347594

ДАМП СО СКРИПТАМИ:

Создаем базу данных:

```
CREATE DATABASE courses WITH TEMPLATE = template0
ENCODING = 'UTF8' LOCALE = 'Russian_Russia.1251';
ALTER DATABASE courses OWNER TO postgres;
\connect courses
SET statement_timeout = 0;
SET lock_timeout = 0;
SET idle_in_transaction_session_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SELECT pg_catalog.set_config('search_path', '', false);
SET check_function_bodies = false;
SET xmloption = content;
SET client_min_messages = warning;
SET row_security = off;
--
-- Name: courses; Type: SCHEMA; Schema: -; Owner: postgres
```

Создаем схему:

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

Создаем таблицы:

```
--
-- Name: classroom ; Type: TABLE; Schema: courses; Owner: postgres
--

CREATE TABLE courses."classroom " (
    classroom_number integer NOT NULL,
    classrom_capacity integer NOT NULL,
    classrom_type character varying(40) NOT NULL,
```

```
    venue_name character varying(40) NOT NULL
);


ALTER TABLE courses."classroom " OWNER TO postgres;

--
-- Name: disciplines; Type: TABLE; Schema: courses; Owner: postgres
--

CREATE TABLE courses.disciplines (
    discipline_id integer NOT NULL,
    programm_id integer NOT NULL,
    discipline_name character varying(40) NOT NULL,
    practice_hours integer NOT NULL,
    lecture_hours integer NOT NULL,
    laboratory_hours integer NOT NULL
);


ALTER TABLE courses.disciplines OWNER TO postgres;

--
-- Name: group; Type: TABLE; Schema: courses; Owner: postgres
--

CREATE TABLE courses."group" (
    group_code integer NOT NULL,
    group_number character varying(15) NOT NULL,
    programm_id integer NOT NULL,
    education_year integer NOT NULL
);


ALTER TABLE courses."group" OWNER TO postgres;

--
-- Name: learning; Type: TABLE; Schema: courses; Owner: postgres
--

CREATE TABLE courses.learning (
```

```
    since date NOT NULL,
    "until " date NOT NULL,
    "student_id " integer NOT NULL,
    "group_code " integer NOT NULL
);


ALTER TABLE courses.learning OWNER TO postgres;

--
-- Name: professor; Type: TABLE; Schema: courses; Owner: postgres
--

CREATE TABLE courses.professor (
    professor_id integer NOT NULL,
    "full_name " character varying(50) NOT NULL,
    "position" character varying(30) NOT NULL
);


ALTER TABLE courses.professor OWNER TO postgres;

--
-- Name: programm; Type: TABLE; Schema: courses; Owner: postgres
--

CREATE TABLE courses.programm (
    programm_id integer NOT NULL,
    "programm_name " character varying(50) NOT NULL
);


ALTER TABLE courses.programm OWNER TO postgres;

--
-- Name: schedule; Type: TABLE; Schema: courses; Owner: postgres
--

CREATE TABLE courses.schedule (
    "class_number " integer NOT NULL,
    "class_type " character varying(30) NOT NULL,
```

```
    "classrom_type " character varying(30) NOT NULL,
    "event_status " text NOT NULL,
    "professor_id " integer NOT NULL,
    "classroom_number " integer NOT NULL,
    "discipline_id " integer NOT NULL,
    "group_code " integer NOT NULL,
    "class_date " character varying(20) NOT NULL,
    "class_start_time " character varying(10) NOT NULL,
    "class_end_time " character varying(10) NOT NULL
);


ALTER TABLE courses.schedule OWNER TO postgres;

--
-- Name: student; Type: TABLE; Schema: courses; Owner: postgres
--

CREATE TABLE courses.student (
    "passport_data " character varying(40) NOT NULL,
    "full_name " character varying(40) NOT NULL,
    "student_id " integer NOT NULL
);


ALTER TABLE courses.student OWNER TO postgres;

--
-- Name: venue; Type: TABLE; Schema: courses; Owner: postgres
--

CREATE TABLE courses.venue (
    "venue_name " character varying(40) NOT NULL,
    "adress " character varying(40) NOT NULL
);

ALTER TABLE courses.venue OWNER TO postgres;
--
-- Data for Name: classroom ; Type: TABLE DATA; Schema: courses; Owner:
postgres
--
```

Задаем ограничения Checks и Foreign key:

```
COPY courses."classroom " (classroom_number, classrom_capacity, classrom_type, venue_name) FROM stdin;
\.
COPY courses."classroom " (classroom_number, classrom_capacity, classrom_type, venue_name) FROM '$$PATH$$/3366.dat';
--
-- Data for Name: disciplines; Type: TABLE DATA; Schema: courses; Owner: postgres
--
COPY courses.disciplines (discipline_id, programm_id, discipline_name, practice_hours, lecture_hours, laboratory_hours) FROM stdin;
\.
COPY courses.disciplines (discipline_id, programm_id, discipline_name, practice_hours, lecture_hours, laboratory_hours) FROM '$$PATH$$/3367.dat';
--
-- Data for Name: group; Type: TABLE DATA; Schema: courses; Owner: postgres
--
COPY courses."group" (group_code, group_number, programm_id, education_year) FROM stdin;
\.
COPY courses."group" (group_code, group_number, programm_id, education_year) FROM '$$PATH$$/3368.dat';
--
-- Data for Name: learning; Type: TABLE DATA; Schema: courses; Owner: postgres
--
COPY courses.learning (since, "until ", "student_id ", "group_code ") FROM stdin;
\.
COPY courses.learning (since, "until ", "student_id ", "group_code ") FROM '$$PATH$$/3369.dat';
--
-- Data for Name: professor; Type: TABLE DATA; Schema: courses; Owner: postgres
--
COPY courses.professor (professor_id, "full_name ", "position") FROM stdin;
\.
COPY courses.professor (professor_id, "full_name ", "position") FROM '$$PATH$$/3370.dat';
```

```
--
-- Data for Name: programm; Type: TABLE DATA; Schema: courses; Owner:
postgres
--
COPY courses.programm (programm_id, "programm_name ") FROM stdin;
\.
COPY courses.programm (programm_id, "programm_name ") FROM
'$$PATH$$/3371.dat';
--
-- Data for Name: schedule; Type: TABLE DATA; Schema: courses; Owner: postgres
--
COPY courses.schedule ("class_number ", "class_type ", "classrom_type ",
"event_status ", "professor_id ", "classroom_number ", "discipline_id ", "group_code
", "class_date ", "class_start_time ", "class_end_time ") FROM stdin;
\.
COPY courses.schedule ("class_number ", "class_type ", "classrom_type ",
"event_status ", "professor_id ", "classroom_number ", "discipline_id ", "group_code
", "class_date ", "class_start_time ", "class_end_time ") FROM '$$PATH$$/3372.dat';
--
-- Data for Name: student; Type: TABLE DATA; Schema: courses; Owner: postgres
--
COPY courses.student ("passport_data ", "full_name ", "student_id ") FROM stdin;
\.
COPY courses.student ("passport_data ", "full_name ", "student_id ") FROM
'$$PATH$$/3373.dat';


--
-- Data for Name: venue; Type: TABLE DATA; Schema: courses; Owner: postgres
--
COPY courses.venue ("venue_name ", "adress ") FROM stdin;
\.
COPY courses.venue ("venue_name ", "adress ") FROM '$$PATH$$/3374.dat';
--
-- Name: professor 3f; Type: CHECK CONSTRAINT; Schema: courses; Owner:
postgres
--
ALTER TABLE courses.professor
    ADD CONSTRAINT "3f" CHECK ((professor_id > 0)) NOT VALID;
--
-- Name: programm 3f; Type: CHECK CONSTRAINT; Schema: courses; Owner:
postgres
```

```
--
ALTER TABLE courses.programm
    ADD CONSTRAINT "3f" CHECK ((programm_id > 0)) NOT VALID;
--
-- Name: group 4f3effff; Type: CHECK CONSTRAINT; Schema: courses; Owner:
postgres
--
ALTER TABLE courses."group"
    ADD CONSTRAINT "4f3effff" CHECK ((group_code > 0)) NOT VALID;
--
-- Name: classroom  classroom _pkey; Type: CONSTRAINT; Schema: courses;
Owner: postgres
--
ALTER TABLE ONLY courses."classroom "
    ADD CONSTRAINT "classroom _pkey" PRIMARY KEY (classroom_number);
--
-- Name: disciplines disciplines_pkey; Type: CONSTRAINT; Schema: courses;
Owner: postgres
--
ALTER TABLE ONLY courses.disciplines
    ADD CONSTRAINT disciplines_pkey PRIMARY KEY (discipline_id);
--
-- Name: disciplines erf; Type: CHECK CONSTRAINT; Schema: courses; Owner:
postgres
--
ALTER TABLE courses.disciplines
    ADD CONSTRAINT erf CHECK ((discipline_id > 0)) NOT VALID;
--
-- Name: group group_pkey; Type: CONSTRAINT; Schema: courses; Owner:
postgres
--
ALTER TABLE ONLY courses."group"
    ADD CONSTRAINT group_pkey PRIMARY KEY (group_code);
--
-- Name: learning learning_pkey; Type: CONSTRAINT; Schema: courses; Owner:
postgres
--
ALTER TABLE ONLY courses.learning
    ADD CONSTRAINT learning_pkey PRIMARY KEY ("student_id ", "group_code
");
--
```

```
-- Name: professor professor_pkey; Type: CONSTRAINT; Schema: courses; Owner: postgres
--
ALTER TABLE ONLY courses.professor
    ADD CONSTRAINT professor_pkey PRIMARY KEY (professor_id);
--
-- Name: programm programm_pkey; Type: CONSTRAINT; Schema: courses; Owner: postgres
--
ALTER TABLE ONLY courses.programm
    ADD CONSTRAINT programm_pkey PRIMARY KEY (programm_id);
--
-- Name: classroom  ref; Type: CHECK CONSTRAINT; Schema: courses; Owner: postgres
--
ALTER TABLE courses."classroom "
    ADD CONSTRAINT ref CHECK ((classroom_number > 0)) NOT VALID;
--
-- Name: schedule schedule_pkey; Type: CONSTRAINT; Schema: courses; Owner: postgres
--
ALTER TABLE ONLY courses.schedule
    ADD CONSTRAINT schedule_pkey PRIMARY KEY ("discipline_id ", "professor_id ", "group_code ", "class_number ", "classroom_number ", "class_date ");
--
-- Name: student student_pkey; Type: CONSTRAINT; Schema: courses; Owner: postgres
--
ALTER TABLE ONLY courses.student
    ADD CONSTRAINT student_pkey PRIMARY KEY ("student_id ");
--
-- Name: venue venue_pkey; Type: CONSTRAINT; Schema: courses; Owner: postgres
--
ALTER TABLE ONLY courses.venue
    ADD CONSTRAINT venue_pkey PRIMARY KEY ("venue_name ");
--
-- Name: schedule 34f; Type: FK CONSTRAINT; Schema: courses; Owner: postgres
--
ALTER TABLE ONLY courses.schedule
```

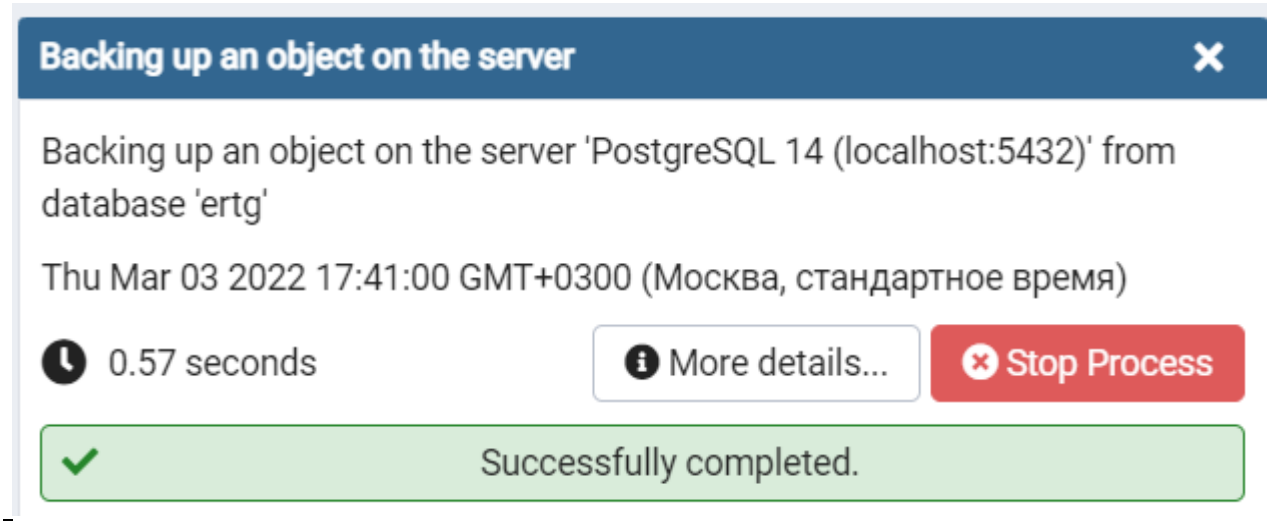
```
    ADD CONSTRAINT "34f" FOREIGN KEY ("group_code ") REFERENCES
courses."group"(group_code) ON DELETE CASCADE DEFERRABLE INITIALLY
DEFERRED;
--
-- Name: classroom  3f4fr; Type: FK CONSTRAINT; Schema: courses; Owner:
postgres
--
ALTER TABLE ONLY courses."classroom "
    ADD CONSTRAINT "3f4fr" FOREIGN KEY (venue_name) REFERENCES
courses.venue("venue_name ") ON DELETE CASCADE DEFERRABLE
INITIALLY DEFERRED;
--
-- Name: group 3rf; Type: FK CONSTRAINT; Schema: courses; Owner: postgres
--
ALTER TABLE ONLY courses."group"
    ADD CONSTRAINT "3rf" FOREIGN KEY (programm_id) REFERENCES
courses.programm(programm_id) ON DELETE CASCADE DEFERRABLE
INITIALLY DEFERRED;
--
-- Name: schedule 43f; Type: FK CONSTRAINT; Schema: courses; Owner: postgres
--
ALTER TABLE ONLY courses.schedule
    ADD CONSTRAINT "43f" FOREIGN KEY ("professor_id ") REFERENCES
courses.professor(professor_id) ON DELETE CASCADE DEFERRABLE
INITIALLY DEFERRED;
--
-- Name: learning ef; Type: FK CONSTRAINT; Schema: courses; Owner: postgres
--
ALTER TABLE ONLY courses.learning
    ADD CONSTRAINT ef FOREIGN KEY ("student_id ") REFERENCES
courses.student("student_id ") ON DELETE CASCADE DEFERRABLE INITIALLY
DEFERRED;
--
-- Name: disciplines f34; Type: FK CONSTRAINT; Schema: courses; Owner:
postgres
--
ALTER TABLE ONLY courses.disciplines
    ADD CONSTRAINT f34 FOREIGN KEY (programm_id) REFERENCES
courses.programm(programm_id) ON DELETE CASCADE DEFERRABLE
INITIALLY DEFERRED;
```

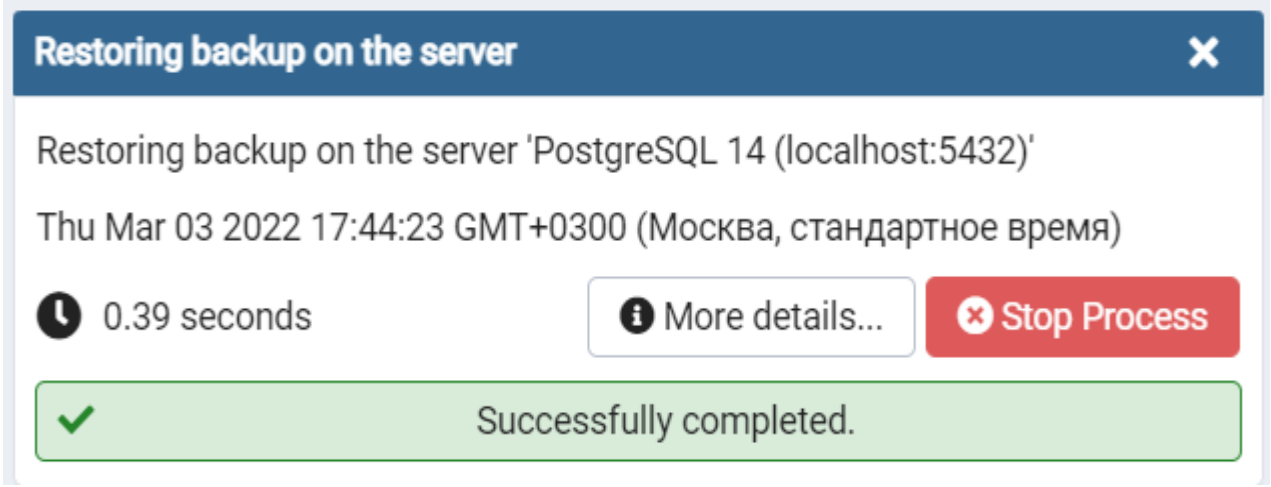
```
--
-- Name: learning rve; Type: FK CONSTRAINT; Schema: courses; Owner: postgres
--

ALTER TABLE ONLY courses.learning
    ADD CONSTRAINT rve FOREIGN KEY ("group_code ") REFERENCES
courses."group"(group_code) ON DELETE CASCADE DEFERRABLE INITIALLY
DEFERRED;



--
-- Name: schedule rve; Type: FK CONSTRAINT; Schema: courses; Owner: postgres
--

ALTER TABLE ONLY courses.schedule
    ADD CONSTRAINT rve FOREIGN KEY ("classroom_number ") REFERENCES
courses."classroom "(classroom_number) ON DELETE CASCADE DEFERRABLE
INITIALLY DEFERRED;



--
-- Name: schedule v; Type: FK CONSTRAINT; Schema: courses; Owner: postgres
--

ALTER TABLE ONLY courses.schedule
    ADD CONSTRAINT v FOREIGN KEY ("discipline_id ") REFERENCES
courses.disciplines(discipline_id) ON DELETE CASCADE DEFERRABLE
INITIALLY DEFERRED;



--
-- Name: DATABASE courses; Type: ACL; Schema: -; Owner: postgres
--

REVOKE ALL ON DATABASE courses FROM postgres;
GRANT CREATE,CONNECT ON DATABASE courses TO postgres;
GRANT TEMPORARY ON DATABASE courses TO postgres WITH GRANT
OPTION;



--
```

-- Name: SCHEMA courses; Type: ACL; Schema: -; Owner: postgres
--

REVOKE ALL ON SCHEMA courses FROM postgres;
GRANT ALL ON SCHEMA courses TO postgres WITH GRANT OPTION;

Созданием бэкапа и его восстановление:





Вывод:

PgAdmin – достаточно удобная программа для создания баз данных PostgreSQL, обладающая приемлемо интуитивным интерфейсом, разобраться с которым новичку не доставит великих проблем. Но, к сожалению, программа обладает неявными ограничениями или даже багами, с которыми новичку самостоятельно справиться будет гораздо тяжелее. К примеру, с чем столкнулся Я: невозможность задать ограничение для столбца, если его имя содержит символы верхнего регистра, необходимость использовать скрипты SELECT, INSERT, DELETE, etc., так как программа не воспринимает стандартный метод ввода SQL.