Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

ITMO University

Отчет по лабораторной работе № 3

По дисциплине Проектирование баз данных/Базы данных

Обучающиеся Наус Михаил Романович, Савинова Алина Константиновна

Факультет Факультет технологического менеджмента и инноваций

Группа U3275

Направление подготовки 27.03.05 Инноватика

Образовательная программа Технологии и инновации

Обучающиеся Наус М. Р. Савинова А. К.

Преподаватель Ромакина О.М.

Основная часть

Цель работы: овладеть практическими навыками создания таблиц базы данных PostgreSQL 10 (11), заполнения их рабочими данными, резервного копирования и восстановления БД.

Практическое задание:

1. Создать базу данных с использованием pgadmin4 (согласно индивидуальному заданию).
2. Создать схему в составе базы данных.
3. Создать таблицы базы данных.
4. Заполнить таблицы БД рабочими данными.
5. Создать резервную копию БД.
6. Восстановить БД на другом ПК.

1. Создать базу данных с использованием pgadmin4.

Для создания базы данных был использован pgadmin4. Для объявления новой базы данных в рамках сервера, мы должны выполнить следующий порядок действий: Нажатие правой кнопки мыши на базы данных сервера – «Create…» - «Database…». После чего указываем имя нашей базы данных в поле – «Database», по желанию можно указать комментарий для этой базы данных.

2. Создать схему в составе базы данных.

Для создания схемы в базе данных мы создадим схему для определения организационной рабочей области базы данных (обязательным условием является статус суперпользователя для этой базы данных или иметь привилегию CREATE).

3. Создать таблицы базы данных.

Для создания таблиц в базе данных мы будем использовать диалект PostgreSQL и инструмент в pgadmin4 – “Query Tool”, исполняемый код представлен ниже (см. Листинг-1).

```
-- Table: Сценарист
CREATE TABLE Сценарист (
    ID_S SERIAL PRIMARY KEY,
    Имя_Фамилия VARCHAR(100) NOT NULL,
    Дата_рождения DATE
);

-- Table: Режиссёр
CREATE TABLE Режиссёр (
    ID_P SERIAL PRIMARY KEY,
    Имя_Фамилия VARCHAR(100) NOT NULL,
    Дата_рождения DATE
);

-- Table: Фильм
CREATE TABLE Фильм (
    ID_F SERIAL PRIMARY KEY,
    Название VARCHAR(100) NOT NULL,
    Год_выпуска INT,
    Жанр VARCHAR(50),
    Рейтинг DECIMAL(3, 1),
    Аннотация TEXT,
    Страна_производитель VARCHAR(50)
);

-- Table: Актёр
CREATE TABLE Актёр (
    ID_A SERIAL PRIMARY KEY,
    Имя_Фамилия VARCHAR(100) NOT NULL,
    Дата_рождения DATE
);

-- Table: Награды
CREATE TABLE Награды (
    ID_H SERIAL PRIMARY KEY,
    Название VARCHAR(100) NOT NULL
);

-- Table: Удостоились
CREATE TABLE Удостоились (
    ID_U SERIAL PRIMARY KEY,
    ID_F INT REFERENCES Фильм(ID_F) ON DELETE CASCADE,
    ID_H INT REFERENCES Награды(ID_H) ON DELETE CASCADE,
    Лауреат VARCHAR(100),
    Год_получения INT,
```

```
    Категория_награды VARCHAR(50)
);

-- Table: Участвовал_в_создании
CREATE TABLE Участвовал_в_создании (
    ID_UC SERIAL PRIMARY KEY,
    ID_S INT REFERENCES Сценарист(ID_S) ON DELETE CASCADE,
    ID_P INT REFERENCES Режиссёр(ID_P) ON DELETE CASCADE,
    ID_F INT REFERENCES Фильм(ID_F) ON DELETE CASCADE
);

-- Table: Роль_в_кино
CREATE TABLE Роль_в_кино (
    ID_R SERIAL PRIMARY KEY,
    ID_F INT REFERENCES Фильм(ID_F) ON DELETE CASCADE,
    ID_A INT REFERENCES Актёр(ID_A) ON DELETE CASCADE,
    Роль VARCHAR(100)
);
```

(Листинг-1)

4. Заполнить таблицы БД рабочими данными.

Для заполнения была также использована лексика и язык запросов PostgreSQL, после чего были загружены соответствующие тестовые рабочие данные в таблицы базы данных. Ниже можно ознакомиться с кодом запросов (см. Листинг-2).

```
-- Insert value for table have only primary key

-- Сценарист
INSERT INTO Сценарист (Имя_Фамилия, Дата_рождения) VALUES
('Иван Иванов', '1970-05-12'),
('Алексей Петров', '1985-03-23'),
('Мария Семёнова', '1990-11-17'),
('Дмитрий Ковалёв', '1978-08-05');

-- Режиссёр
INSERT INTO Режиссёр (Имя_Фамилия, Дата_рождения) VALUES
('Сергей Смирнов', '1968-11-30'),
('Ольга Кузнецова', '1980-07-18'),
('Александр Белов', '1975-04-10'),
('Наталья Воронова', '1983-06-22');

-- Фильм
INSERT INTO Фильм (Название, Год_выпуска, Жанр, Рейтинг, Аннотация, Страна_производитель)
VALUES
```

```
('Фильм 1', 2022, 'Драма', 8.5, 'Описание фильма 1', 'Россия'),
('Фильм 2', 2021, 'Комедия', 7.3, 'Описание фильма 2', 'Россия'),
('Фильм 3', 2020, 'Триллер', 6.9, 'Описание фильма 3', 'США'),
('Фильм 4', 2019, 'Боевик', 8.0, 'Описание фильма 4', 'Канада');

-- Актёр
INSERT INTO Актёр (Имя_Фамилия, Дата_рождения) VALUES
('Мария Павлова', '1992-01-15'),
('Андрей Сидоров', '1983-09-09'),
('Екатерина Орлова', '1995-04-02'),
('Владимир Нестеров', '1980-12-19');

-- Награды
INSERT INTO Награды (Название) VALUES
('Оскар'),
('Золотой глобус'),
('BAFTA'),
('Каннский фестиваль');
```

(Листинг-2.1 «Без внешних ключей»)

```
-- For table have reference (reference key)

-- Удостоились
INSERT INTO Удостоились (ID_F, ID_H, Лауреат, Год_получения, Категория_награды) VALUES
(1, 1, 'Фильм 1', 2023, 'Лучший фильм'),
(2, 2, 'Фильм 2', 2022, 'Лучшая комедия'),
(3, 3, 'Фильм 3', 2021, 'Лучший сценарий'),
(4, 4, 'Фильм 4', 2020, 'Лучший режиссёр'),
(1, 2, 'Фильм 1', 2023, 'Лучшая драма'),
(2, 3, 'Фильм 2', 2022, 'Лучший актёр'),
(3, 1, 'Фильм 3', 2021, 'Лучший триллер'),
(4, 4, 'Фильм 4', 2020, 'Лучший боевик');

-- Участвовал_в_создании
INSERT INTO Участвовал_в_создании (ID_S, ID_P, ID_F) VALUES
(1, 1, 1),
(2, 2, 2),
(3, 3, 3),
(4, 4, 4),
(1, 2, 3),
(3, 1, 4),
(2, 3, 1),
(4, 2, 2);

-- Роль_в_кино
INSERT INTO Роль_в_кино (ID_F, ID_A, Роль) VALUES
(1, 1, 'Главная роль'),
(1, 2, 'Второстепенная роль'),
(2, 3, 'Главная роль'),
(2, 4, 'Второстепенная роль'),
(3, 1, 'Эпизодическая роль'),
```

```
(3, 3, 'Главная роль'),
(4, 2, 'Роль второго плана'),
(4, 4, 'Главная роль');
```

(Листинг-2.2 «С внешними ключами»)

Также в рамках создания, внесения данных в таблицы базы данных, хочется показать и обновление определенных данных и их удаление. (см. Листниг-3)

```
-- Example for DELETE and UPDATE

UPDATE Фильм
SET Рейтинг = 9.0
WHERE Название = 'Фильм 1';

UPDATE Удостоились
SET Категория_награды = 'Лучший актёр'
WHERE Лауреат = 'Фильм 3' AND Год_получения = 2021;

UPDATE Роль_в_кино
SET Роль = 'Главная роль'
WHERE ID_A = 2 AND ID_F = 4;

DELETE FROM Фильм
WHERE Название = 'Фильм 2';

DELETE FROM Актёр
WHERE Имя_Фамилия = 'Владимир Нестеров';

DELETE FROM Удостоились
WHERE ID_F = 1 AND ID_H = 2;
```

(Листниг-3 «Обновление и удаление данных»)

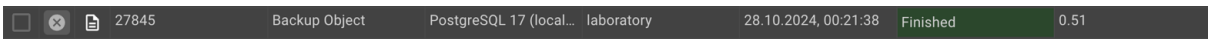
5. Создать резервную копию БД.

Для создания резервной копии базы данных мы должны произвести ряд действий: Нажать правой кнопкой по базе данных – «Backup…». Далее нам требуется указать имя, мы также можем указать путь для сохранения нашей резервной копии. В нашем случае мы указали следующие характеристики:

1. Имя: laboratory_backup (Также задали ему путь сохранения)

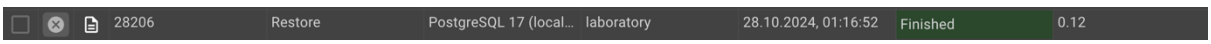
2. Format: Custom
3. Compression ratio: 9
4. Encoding: WIN1251
5. Number of jobs: ----
6. Role name: postgres (то есть суперпользователь)

После чего была успешно создана резервная копия:



6. Восстановить БД на другом ПК.

Для восстановления на другом компьютере мы использовали файл, который ранее создали. После чего заходим в pgadmin4, выбираем наш сервер и далее требуется база данных, для которой мы будем восстанавливать данные. В нашем случае оно несет точно такое же название как и предыдущая база данных. Нажимаем правой кнопкой мыши – «Restore…». Указываем файл с резервной копией, после чего количество потоков для восстановления таблиц базы данных и, конечно, роль. После чего можем наблюдать успешное восстановление базы данных:



Вывод

В ходе работы мы познакомились с созданием базы данных PostgreSQL, средствами pgAdmin. Было создано несколько таблиц на основе диаграммы IDE1FX, то есть была дана практическая ее реализация. Научились работать с вставкой информации в таблицы базы данных, её обновлением или удалением. Вместе с тем научились создавать резервные копии базы данных и восстанавливать с помощью неё рабочую версию на другом ПК. Со всеми исходными данными и кодами вы можете ознакомиться на github одного из членов команды: https://github.com/fallayn/Database_Laboratory_No_3.git