Лабораторная работа №3: Создание базы данных

---

1. Устанавливаем PostgreSQL
   Установите пакет Postgres вместе с пакетом -contrib, который содержит дополнительные утилиты и функциональные возможности:
   ➔ **sudo apt install postgresql postgresql-contrib**
   По умолчанию Postgres использует концепцию «ролей» для выполнения аутентификации и авторизации.
   В ходе установки была создана учетную запись пользователя postgres, которая связана с используемой по умолчанию ролью postgres. Существует несколько способов использования этой учетной записи для доступа к Postgres. Один из способов — переход к учетной записи postgres на вашем сервере с помощью следующей команды:
   ➔ **sudo -i -u postgres**
   Затем вы можете получить доступ к командной строке Postgres с помощью команды:
   ➔ **psql**

2. Создание базы данных
   Чтобы выполнять базовые действия в СУБД, нужно знать Structured Query Language (SQL).
   Для создания базы данных используется команда create database. В приведенном ниже примере создается база данных с именем mtuci_db.

```
CREATE DATABASE mtuci_db
```

Для подключения к созданной базе данных необходимо выполнить команду

```
\c mtuci_db
```

3. **Создание таблиц**
   Таблицы являются объектами, которые содержат все данные в базах данных. В таблицах данные логически организованы в виде строк и столбцов по аналогии с электронной таблицей. Каждая строка представляет собой уникальную запись, а каждый столбец — поле записи. Например, таблица, содержащая данные о сотрудниках компании, может иметь строку для каждого сотрудника и столбцы, представляющие сведения о сотрудниках (например, его идентификационный номер, имя, адрес)
   Вы можете создать таблицу, указав её имя и перечислив все имена столбцов и их типы:

```
CREATE TABLE student_group (id SERIAL PRIMARY KEY, numb varchar NOT NULL, chair varchar NOT NULL)
```

SERIAL - целое число, которое увеличивается на 1 при добавлении новой записи

PRIMARY KEY - ограничение первичного ключа означает, что образующий его столбец или группа столбцов может быть уникальным идентификатором строк в таблице. Для этого требуется, чтобы значения были одновременно уникальными и отличными от NULL.

integer - целые числа

varchar(n) - строка ограниченной переменной длины

NOT NULL - ограничение на недопустимость ввода пустого значения

4. **Добавление записей в таблицу**

   Для добавления данных применяется команда INSERT. После INSERT INTO идет имя таблицы, затем в скобках указываются все столбцы через запятую, в которые надо добавлять данные. И в конце после слова VALUES в скобках перечисляются добавляемые значения:

```
INSERT INTO student_group (numb, chair) VALUES ('БВТ2001', 'МКиИТ')
```

5. **Выборка**

   Для извлечения данных из таблицы используется команда SELECT. Она имеет следующий синтаксис:

```
SELECT список_столбцов FROM имя_таблицы
```

   Если необходимо вывести все столбцы содержащиеся в таблице, вместо перечисления можно использовать *.

   При необходимости получить записи, соответствующие каким-то условиям, следует использовать оператор WHERE, после которого указывается условие, на основании которого производится фильтрация. Например:

```
SELECT chair FROM student_group WHERE numb='БВТ2001';
```

   В PostgreSQL можно применять следующие операции сравнения:

   =: сравнение на равенство

   <>: сравнение на неравенство

   <: меньше чем

   >: больше чем

   !<: не меньше чем

   !>: не больше чем

   <=: меньше чем или равно

   >=: больше чем или равно

6. **Удаление записи**

   Команда DELETE удаляет из указанной таблицы строки, удовлетворяющие условию WHERE. Если предложение WHERE

отсутствует, она удаляет из таблицы все строки, в результате будет получена рабочая, но пустая таблица. Например:

```
DELETE FROM student_group WHERE numb='БВТ2001';
```

Данная команда удаляет все записи, в которых numb='БВТ2001'.

7. **Обновление записи**
   UPDATE изменяет значения указанных столбцов во всех строках, удовлетворяющих условию. В предложении SET должны указываться только те столбцы, которые будут изменены; столбцы, не изменяемые явно, сохраняют свои предыдущие значения. Например:

```
UPDATE student_group SET numb='БИН2005' WHERE chair='СиСС'
```

Данная команда устанавливает значение поля numb равным 'БИН2005' всем записям, в которых поле chair равно 'СиСС'.

8. **Связи между таблицами**
   Связь между таблицами устанавливает отношения между значениями в ключевых полях — часто между полями, имеющими одинаковые имена в обеих таблицах. В большинстве случаев с первичным ключом одной таблицы, являющимся уникальным идентификатором каждой записи, связывается внешний ключ другой таблицы.
   Внешний ключ устанавливается для столбца из зависимой, подчиненной таблицы, и указывает на один из столбцов из главной таблицы.
   Создадим еще одну таблицу, содержащую внешний ключ и свяжем ее с таблицей student_group:

```
CREATE TABLE student (id SERIAL PRIMARY KEY, full_name varchar NOT NULL, passport varchar(10) NOT NULL, group_numb varchar REFERENCES student_group(numb))
```

Домашнее задание:
Создайте следующую базу данных:

1. таблица с информацией о кафедре (id, название, деканат)
2. таблица с информацией о студенческой группе (id, название, кафедра)
3. таблица с информацией о студентах (id, имя, паспортные данные, группа).
4. Между всеми таблицами должны быть связи. Заполнить таблицу кафедра 2 записями, таблицу групп 4 записями (по 2 группы на кафедру) и в таблицу студенты по 5 студентов на группу.