# Введение

Настоящее пособие содержит методические указания к выполнению лабораторных работ по дисциплине «Базы данных». Пособие адресовано студентам кафедры АСУ ТУСУР.

Цель лабораторного практикума – освоить основные приёмы реализации реляционных баз данных и компонентов приложений.

Практикум состоит из четырёх лабораторных работ (ЛР). На весь цикл студент получает одно индивидуальное задание (проект), содержащее описание структуры базы данных и функций приложения пользователя. В ходе выполнения ЛР этот проект должен быть реализован. Работы выполняются с использованием реляционной СУБД PostgreSQL. Индивидуальные задания выдаются преподавателем.

В конце каждого занятия следует скопировать текущее состояние проекта (.dump-файл) в директорию на сетевом диске. Подробнее о том как получить .dump-файл сказано в описании 1 лабораторной работы.

Отчёт о лабораторной работе необходимо сдать преподавателю до начала следующего занятия. Отчёт должен быть оформлен в соответствии с требованиями стандарта «Стандарт ОС ТУСУР 01-2021 по направлениям подготовки и специальностям технического профиля». На титульном листе отчёта обязательно должна быть личная подпись автора и дата представления. Ниже описаны отдельные лабораторные работы цикла и сформулированы требования к содержанию отчётов.

# Общие сведения о СУБД PostgreSQL

### Принцип работы

В книге «Postgres: Первое знакомство» [1] для PostgreSQL дано следующее определение:

«*PostgreSQL — наиболее полнофункциональная, свободно распространяемая СУБД с открытым кодом. Разработанная в академической среде, за долгую историю сплотившая вокруг себя широкое сообщество разработчиков, эта СУБД обладает всеми возможностями, необходимыми большинству заказчиков*.»

С этим определением сложно не согласиться, действительно, на сегодняшний день, PostgreSQL имеет очень большой набор встроенных возможностей, а так же некоторое количество дополнительно подключаемых расширений, которые позволяют использовать СУБД в проектах разной направленности и сложности.

**Заметка**

Кстати, произносить название СУБД следует как «постгрес-ку-эль» или просто «постгрес», но только не «постгре».

Работа с PostgreSQL подразумевает подключение к *серверу PostgreSQL.* Сам сервер при этом является особой программой, которая запущена в фоновом (т.е. неинтерактивном) режиме. Во время своей работы сервер всё время находится в режиме ожидания. Он ожидает запросов на подключение от *клиентов*, которые тоже являются программами. Программа *клиент* может управляться человеком, например, через графическую оболочку, а может выполнять действия согласно некоторому алгоритму, который реализован в её исходном коде. В любом случае, получив запрос на подключение, сервер сначала проводит процедуру аутентификации подключаемого клиента и только после этого начинает принимать и выполнять управляющие команды.

Команды, которые можно отправить на сервер PostgreSQL представляют собой запросы на языке SQL. Получив такой запрос, сервер его анализирует, проверяет имеет ли клиент права на выполнение подобного запроса, проводит оптимизацию и, наконец, выполняет, возвращая результат запроса обратно клиенту. Так как язык SQL является универсальным для всех реляционных СУБД, команды-запросы, сформулированные для сервера PostgreSQL будут мало отличаться от подобных же команд, построенных для другой СУБД, например MySQL.

Однако кроме SQL команд, сервер PostgreSQL может принимать некоторый набор инструкций, действительных только для СУБД PostgreSQL. В большинстве своём это команды каким-то образом управляющие поведением сервера. Подробнее о таких командах Вы узнаете в тексте описания 1 лабораторной работы.

Наглядно схема взаимодействия клиентской программы и сервера СУБД представлена на рисунке 2.1 на следующей странице.

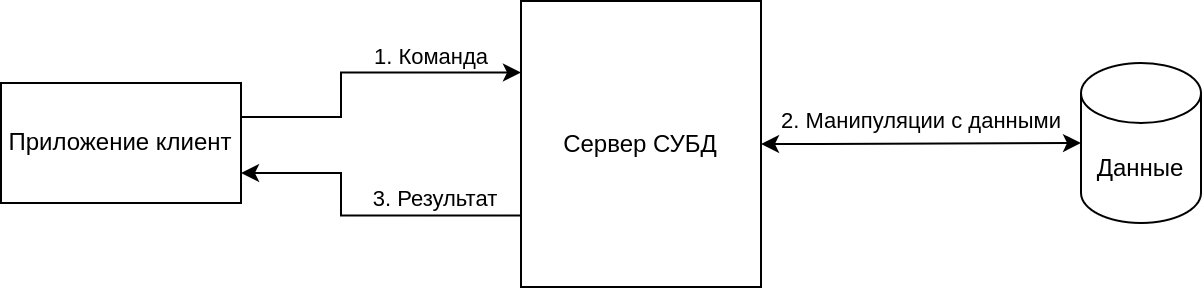


Рисунок 2.1 - Схема взаимодействия клиентского приложения с сервером СУБД

### Структура данных

Любой сервер СУБД управляет в первую очередь данными. Организация этих данных в разных СУБД выполняется в целом одинаково, но есть и различия. Вот основные структурные единицы представления данных в PostgreSQL:

* Кластер баз данных – совокупность всех баз данных, которые имеются на сервере.
* База данных – совокупность именованных схем базы данных.
* Схема базы данных – совокупность логически объединенных отношений, типов данных, функций и операторов.
* Отношение – таблица базы данных, индекс, представление, материализованное представление.

В более наглядном виде структура данных сервера PostgreSQL представлена на рисунке 2.2.

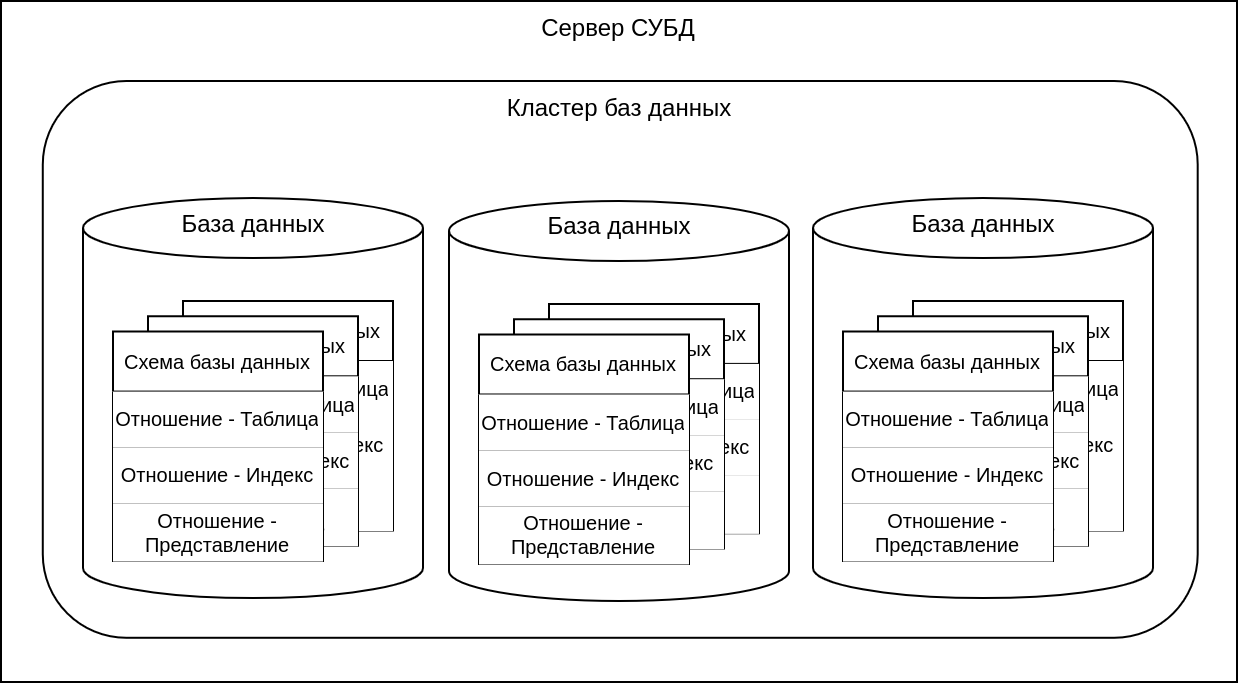


Рисунок 2.1 - Структура данных СУБД PostgreSQL

# Предварительная подготовка

Студенты, выполняющие лабораторные работы с использованием компьютеров кафедры, должны убедиться, что на их рабочем месте установлен сервер PostgreSQL, а так же необходимый к нему инструментарий. Для этого достаточно проверить на наличие приложения «psql.exe»в меню «Пуск». Если такого приложения нет – следует обратиться к преподавателю.

Те же, кто хочет выполнять задания на своих устройствах должны выполнить установку PostgreSQL. Установщик можно найти на официальном сайте:

<https://www.postgresql.org/download/>

Подробная инструкция по установке представлена в книге «Postgres: Первое знакомство»[1], которая находится в открытом доступе на сайте:

<https://www.postgrespro.ru/education/books/introbook>

В случае возникновения проблем, можете обращаться к преподавателю.

!!! ДОБАВИТЬ ПРО КОДИРОВКУ В ОС WINDOWS !!!

# Описания лабораторных работ

## Лабораторная работа №1

### Общие сведения о таблицах и связях

#### Что такое таблица

**Определение**

Таблицей в большинстве реляционных СУБД называется объект базы данных, представляющий собой совокупность строк и столбцов, на пересечении которых размещаются значения простых типов данных.

При этом:

* Все значения столбца принадлежат одному домéну, т.е. имеют один тип данных и удовлетворяют одному набору ограничений.
* Каждая таблица имеет уникальное имя и сохраняемое в системном каталоге определение (макет).
* Макет можно понимать как список имён столбцов таблицы (полей) с указанием их свойств.
* Данные не упорядочены (в частности, нельзя полагаться на то, что строки хранятся в порядке их добавления в таблицу).

#### Для чего предназначена таблица

В таблице сохраняются сведения, относящиеся к одной и только одной теме (объекту или факту). Например:

* таблица СТУДЕНТ содержит личные данные студентов;
* таблица ДИСЦИПЛИНА содержит сведения об учебных дисциплинах, изучаемых студентами;
* таблица УСПЕВАЕМОСТЬ содержит сведения об оценках, полученных студентами на экзаменах (фактах сдачи экзаменов) и т.п.

**Замечание**

Таблица **СТУДЕНТ** может содержать поля *Номер\_студбилета*, *Фамилия*, *Имя*, *Отчество*, *Группа* и не может содержать поле *Оценка*. Это поле характеризует успеваемость студента и должно быть определено в таблице **УСПЕВАЕМОСТЬ**.

#### Что такое первичный ключ таблицы

**Определение**

Для каждой таблицы должен быть определён первичный ключ. Первичный ключ – это поле (группа полей), значения которого идентифицируют строки таблицы. Первичный ключ может быть простым и составным.

На рисунке 4.1 представлены примеры простого и составного первичных ключей.

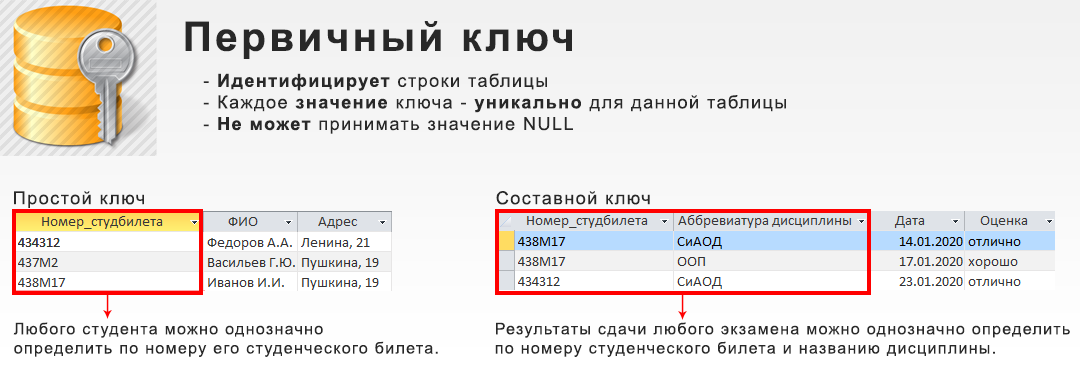


Рисунок 4.1 - Первичный ключ

Первичный ключ обладает тремя важными свойствами:

1. Ни при каких условиях таблица не может содержать двух строк с одинаковыми значениями первичного ключа – свойство уникальнозначности.
2. Составной первичный ключ не содержит подмножества полей, обладающего свойством уникальнозначности – свойство неизбыточности.
3. Первичный ключ не может принимать неопределённые (NULL) значения.

**Замечание**

Первичным ключом **нельзя** объявить произвольное подмножество полей таблицы.

Заключение о наличии или отсутствии свойств первичного ключа у некоторого подмножества полей можно сделать только исходя из смысла данных, сохраняемых в таблице.

Примеры правильного и неправильного выбора первичного ключа можно увидеть на рисунках 4.2 и 4.3 (на следующей странице).

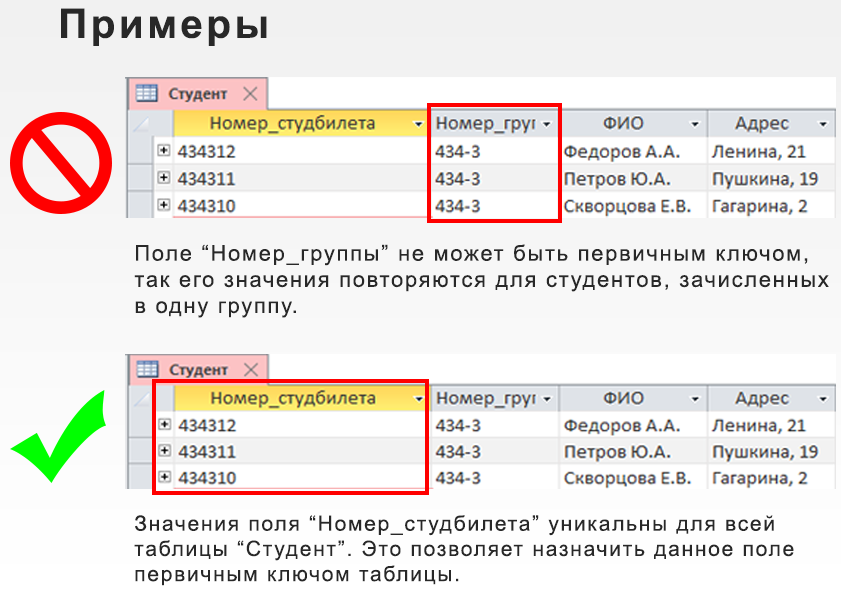


Рисунок 4.2 - Пример назначения первичных ключей

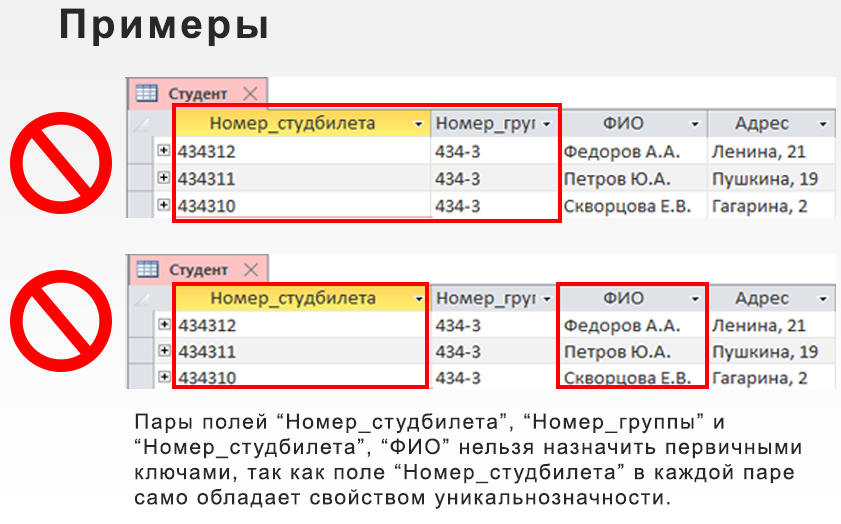


Рисунок 4.3 - Пример назначения составных первичных ключей

#### Для чего нужен первичный ключ

Поскольку значения первичных ключей идентифицируют строки таблицы, их можно использовать:

* + Для быстрого поиска отдельных строк.
  + Для организации связей таблиц.

**Заметка**

Для ускорения поиска записей все СУБД позволяют создавать индексы – особые объекты схемы базы данных.

СУБД PostgreSQL может предложить следующие типы индексов: B-деревья, Hash, GiST, SP-GiST, GIN, RUM, BRIN, Bloom. Подробнее о каждом типе индекса смотрите в книге «PostgreSQL: первое знакомство» [1].

#### Связи таблиц и внешние ключи

**Определение**

Для организации связей в состав полей таблицы-потомка включается дубликат первичного ключа родителя. Он называется здесь *внешним ключом*.

Значения внешнего ключа – это ссылки на строки родительской таблицы, содержащие такие же значения первичного ключа. Ссылки реализуются как неуникальные индексы по внешнему ключу. Эти индексы хранят информацию о связи, т.е. соответствии строк различных таблиц.

На рисунке 4.4 показаны строки из двух таблиц – «Студент» и «Успеваемость». Видно, что поле «*Номер\_студбилета*» таблицы «Успеваемость» ссылается на такое же поле родительской таблицы.

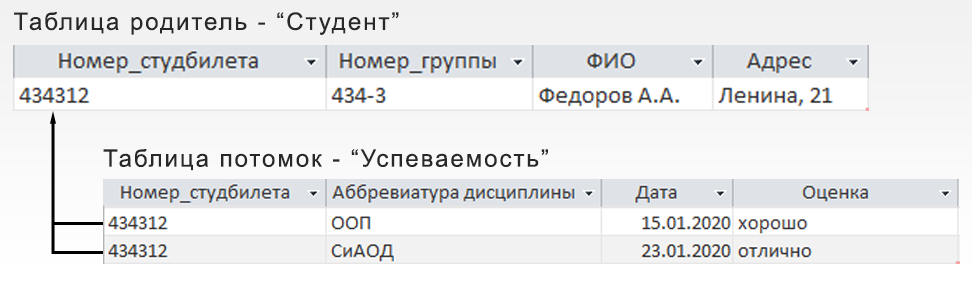


Рисунок 4.4 - Строки из таблиц «Студент» и «Успеваемость»

#### Ссылочная целостность

**Определение**

Значения внешнего ключа в строках таблицы-потомка **не могут** быть произвольными. Они должны принадлежать множеству значений родительского ключа в существующих строках родительской таблицы. Это условие называется требованием *ссылочной целостности*.

Далее на рисунке 4.5 на следующей странице продемонстрирован пример нарушения ссылочной целостности. Строка таблицы «Успеваемость» ссылается на несуществующий номер студенческого билета.

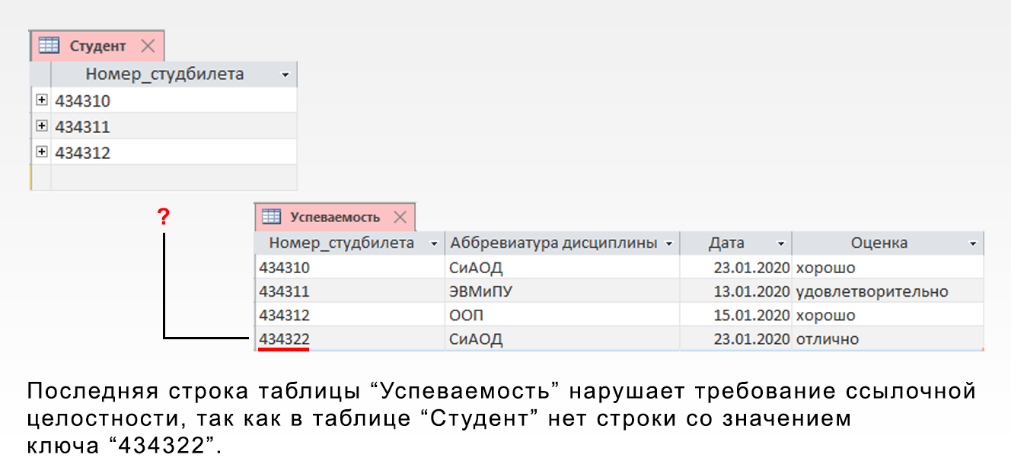


Рисунок 4.5 - Нарушение ссылочной целостности

#### Правила ссылочной целостности

**Замечание**

Одна из главных задач СУБД – поддерживать ссылочную целостность при выполнении операций обновления данных. Она будет это делать, если для каждой пары родитель-потомок определены правила совместного обновления множеств значений первичного и внешнего ключей.

Ссылочная целостность может быть нарушена в следующих случаях:

* + При попытке добавления строки в таблицу-потомок.
  + При попытке изменения значения внешнего ключа в существующей строке таблицы-потомка.
  + При попытке удаления строки из родительской таблицы.
  + При попытке изменения значения первичного ключа в существующей строке родительской таблицы.

В двух первых случаях СУБД должна проверить, принадлежит ли новое значение внешнего ключа множеству существующих значений родительского ключа. Если да, то обновление таблицы-потомка будет выполнено, иначе – отвергнуто.

В двух последних случаях вариантов правил обновления гораздо больше. Они определяются требованиями пользователя и должны быть реализованы разработчиком БД в виде специальных программ – триггеров ссылочной целостности.

### Начало работы

В первой лабораторной работе Вы будете взаимодействовать с сервером СУБД, используя стандартный терминальный клиент - psql.

Отсутствие графического интерфейса может показаться неудобным, однако psql является стандартным приложением, использовать которое можно в любой операционной системе (ОС) и в любой момент. Таким образом, имея навыки работы с psql, Вы сможете взаимодействовать с Вашей базой данных даже если работаете в незнакомой Вам ОС или когда все графические средства оказались недоступными.

#### Запуск psql

**Замечание**

Запуск psql и работа с сервером СУБД может немного отличаться в разных операционных системах.

Текст методического пособия предполагает, что читатель использует ОС Windows.

Для запуска терминального клиента достаточно найти в меню «Пуск» приложение с соответствующим названием:

*SQL Shell (psql)*

После запуска psql предложит Вам ввести пароль пользователя postgresql. Здесь следует указать пароль, который Вы указывали при установке сервера PostgreSQL.

В случае успешной авторизации, Вы должны увидеть приглашающую строку следующего вида:

*postgres=#*

Это значит, что терминальный клиент успешно подключился к серверу PostgreSQL и ожидает ввода команды.

#### Команды psql

Как уже говорилось в вводной части, каждый СУБД сервер имеет некоторый набор команд, которые являются уникальными для каждого конкретного сервера. Такие команды не связаны с SQL. Найти список всех таких команд, можно в официальной документации, здесь же приведены только команды, важные для выполнения лабораторной работы.

**Заметка**

Все команды psql начинаются с символа обратного слэша (\). После него указывается один или несколько символов, которые и являются названием команды.

Если Вы хотите использовать команду, но забыли как она обозначается, всегда можно воспользоваться справкой, которая вызывается с помощью команды:

*\?*

Вот список полезных для нас команд:

|  |  |
| --- | --- |
| \? | выдаёт справку по командам psql |
| \h | выдаёт справку по командам SQL (список доступных на данный момент команд или справка по конкретной команде) |
| \с название\_базы\_данных | выполняет подключение к указанной базе данных |
| \l | выводит список баз данных, доступных для подключения на текущем сервере |
| \dt | выводит список таблиц подключенной базы данных |
| \d имя\_таблицы | выводит описание указанной таблицы |
| \s имя-файла-для-сохранения-истории-команд | сохраняет историю вызова команд в указанный файл |

#### SQL команды

**Замечание**

Данное методическое пособие не пытается быть учебником по SQL, а потому в этом разделе не стоит искать подробнейшее описание всех команд и принципов работы SQL. Подобные описания и теоретический материал представлены в учебнике «PostgreSQL. Основы языка SQL»[2], который представлен в открытом доступе по ссылке:

<https://postgrespro.ru/education/books/sqlprimer>

В рамках выполнения первой лабораторной работы, Вам необходимо будет использовать следующие SQL команды:

|  |  |
| --- | --- |
| CREATE DATABASE | создание новой базы данных |
| CREATE TABLE | создание новой таблицы |
| ALTER TABLE | изменение структуры существующей таблицы |
| DROP TABLE | удаление таблицы |
| INSERT INTO | вставка новых значений в таблицу |
| UPDATE | обновление существующих значений в таблице |
| SELECT FROM | выборка значений из таблицы |
| DELETE FROM | удаление данных из таблицы |

Кроме того, задания предполагают использование следующих инструментов языка SQL:

|  |  |
| --- | --- |
| PRIMARY KEY | создание первичного ключа в таблице |
| REFERENCES\FOREIGN KEY | создание внешнего ключа в таблице |
| CONSTRAINT CHECK | создание именованного ограничения на значения в столбце таблицы |
| SERIAL | создание поля с автоматическим увеличением значения (только числа) |

**Заметка**

Все команды SQL должны завершаться точкой с запятой (;). Клиент psql не будет отправлять команду на сервер до тех пор пока не встретит этот символ.

#### Резервное сохранение результатов работы

Использование сервера СУБД предполагает, что данные будут надежно сохранены и без дополнительных действий со стороны пользователя. Тем не менее важно знать как можно сохранить текущее состояние базы данных в отдельный файл. Особенно важным будет это умение для тех студентов, которые выполняют лабораторные на компьютерах кафедры, где нельзя гарантировать, что кто-то случайно не удалит все ваши труды (например, командой DROP DATABASE).

Текущее состояние базы данных можно сохранить в виде скрипта, то есть в виде набора инструкций SQL. В дальнейшем СУБД, используя эти инструкции, сможет воссоздать сохраненную базу данных вместе с данными, присутствующими в ней на момент создания скрипта.