Министерство науки и образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный технический университет имени Н.Э.

Баумана

(национальный исследовательский университет)»

(МГТУ им. Н.Э. Баумана)



**УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ К ЛАБОРАТОРНЫМ РАБОТАМ**

**ПО КУРСУ «Базы данных (ЮР)»**

**Лабораторная работа № 5** **«****Введение в PostgreSQL. Создание базы данных»**

Авторы:

Кудрявцев А.П., [kudryavtsevap@bmstu.ru](mailto:kudryavtsevap@bmstu.ru)

Фомин М.М.

Ланцберг А.В.

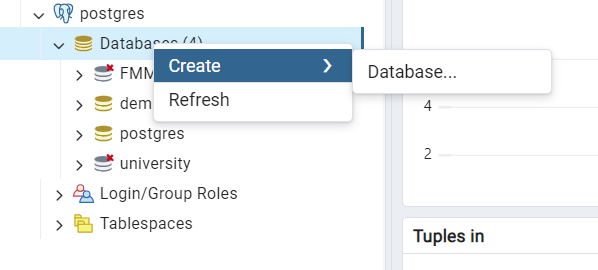
Москва, 2024

## Создание базы данных с помощью pgAdmin

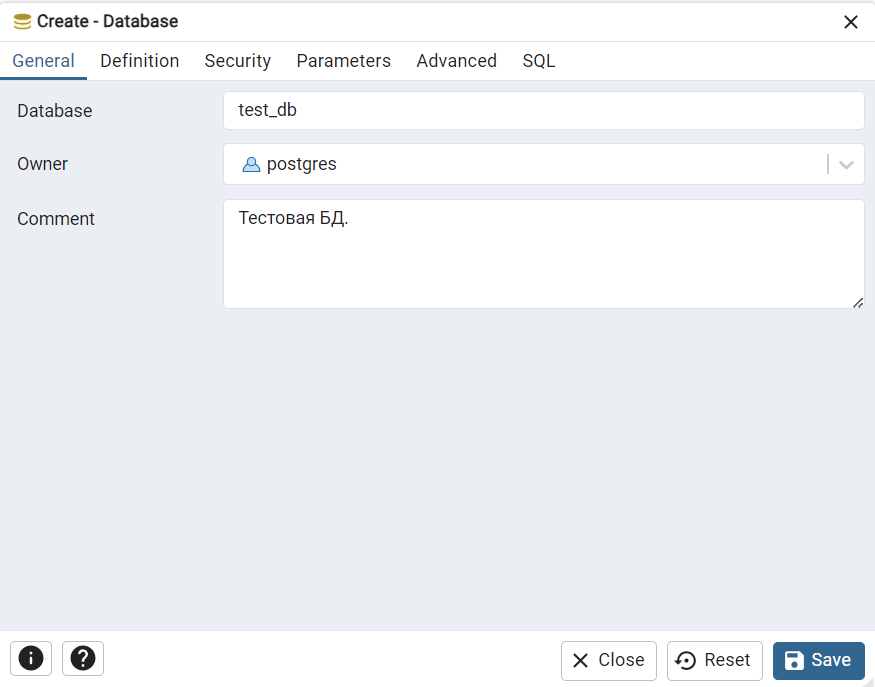
1.1. Создание базы данных

Чтобы создать базу данных, необходимо быть суперпользователем[[1]](#footnote-1) или иметь специальное право CREATEDB.

Откроем pgAdmin. Щелкните правой кнопкой по узлу Databeses и выберите пункт меню *Create/ Database...*.



Появится диалоговое окно для ввода подробной информации о новой базе данных. Диалог *Create – Database* (создание базы данных) имеет следующие вкладки: *General* (общие), *Definition* (определение), *Security* (безопасность), *Parameters* (параметры). Вкладка *SQL* содержит код SQL, который сгенерируется автоматически на основе выбранных параметров в диалоге, и будет выполнен для создания БД.

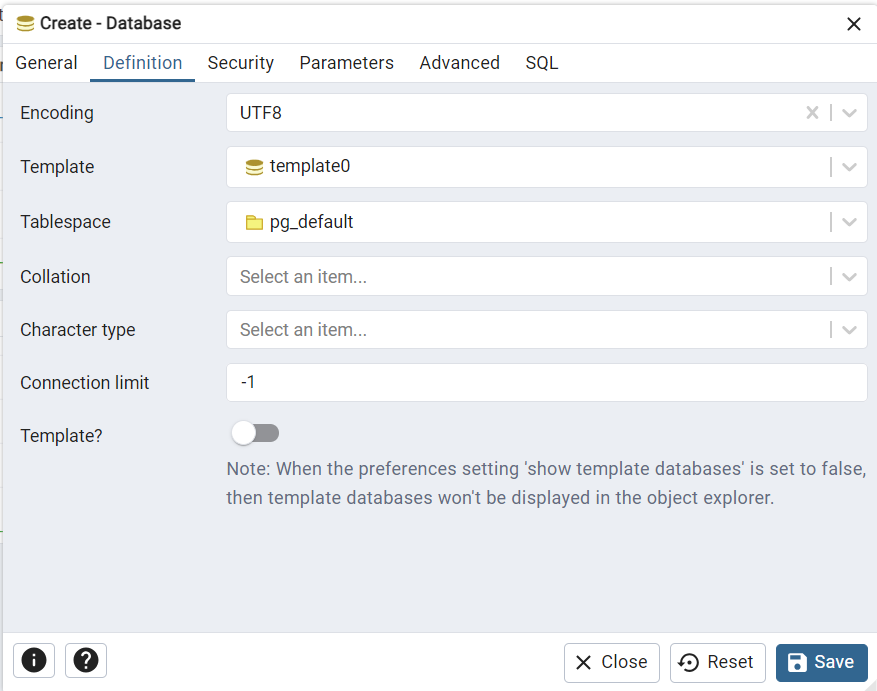


Во вкладке *General* в поле *Database* введем имя БД, пусть это будет, test\_db.

В поле *Owner* выберем из списка владельца БД – postgres.

Заметки можно сохранять в поле *Comment*.

Выберем вкладку *Definition*, чтобы задать свойства базы данных:



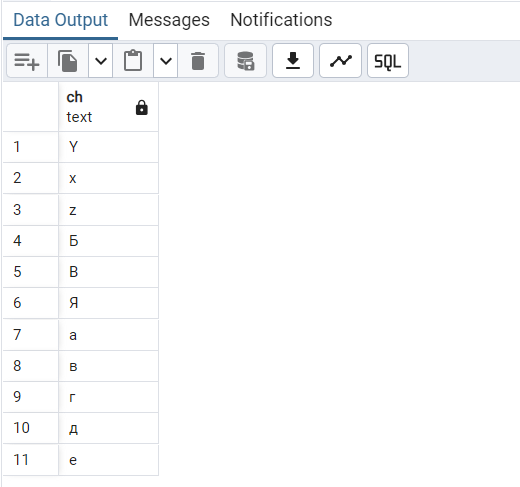
Зададим свойства БД во вкладке *Definition*:

* *Encoding* – кодировка символов в новой базе данных, выбирается из предложенного списка. Оставим по умолчанию UTF8.
* *Template* – имя шаблона, из которого будет создаваться новая база данных, по умолчанию из template1. Если добавить объекты в template1, то впоследствии они будут копироваться в новые базы данных. Это позволяет внести изменения в стандартный набор объектов. Поддерживать template1 имеет смысл, если вам приходится часто создавать похожие БД, содержащие определенный набор таблиц, функций и другие объекты. Также существует вторая системная база template0. В template0 не следует вносить никакие изменения после установки сервера. Если указать в качестве шаблона template0 вместо template1, вы сможете получить «чистую» пользовательскую базу данных (в которой никаких пользовательских объектов нет, есть только системные объекты в первозданном виде), не содержащую ничего, что могло быть добавлено на месте в template1. Другая причина, для копирования template0 вместо template1 заключается в том, что можно указать новые параметры локали и кодировку при копировании template0, в то время как для копий template1 они не должны меняться. Это связано с тем, что template1 может содержать данные в специфических кодировках и локалях, в отличие от template0. Выберем template0.
* *Tablespace* – табличное пространство — это область на диске, где хранятся файлы данных, содержащие объекты базы данных, такие как индексы и таблицы. PostgreSQL использует табличное пространство для связывания логического имени с физическим местоположением на диске. Табличные пространства позволяют вам контролировать структуру диска PostgreSQL. Например, если в разделе диска, на котором был инициализирован кластер, заканчивается место, вы можете создать новое табличное пространство в другом разделе и использовать его до тех пор, пока не перенастроите систему. Вы так же можете разместить индексы или таблицы частого доступа на быстродействующих устройствах, таких как твердотельные накопители, и хранить редко используемые архивные данные на более медленных устройствах. Выберите табличное пространство из раскрывающегося списка – это будет табличное пространство по умолчанию pg\_default, находящееся в каталоге $PGDATA/base (значение $PGDATA можно посмотреть в файле postgresql.conf, для Linux это может быть каталог /var/lib/postgresql/<версия postgresql>/main/).
* *Collation, Character type* – эти два параметра определяют порядок сортировки строк в запросах с использованием ORDER BY или с использованием операторов сравнения, функций upper, lower и др. Выберите из списка *ru\_RU.UTF-8*, если у вас этого значения нет, то *C.UTF-8[[2]](#footnote-2)*. Надо иметь ввиду, что *C.UTF-8* определяет сортировку строк в соответствии с их числовыми кодами, где заглавные буквы имеют меньшие значения, чем строчные. Например, выполним сортировку символов 'а', 'Б', 'в', 'г', 'д', 'е', 'Я', 'z', 'Y', 'x' для БД с кодировкой  
   *C.UTF-8*.

SELECT ch

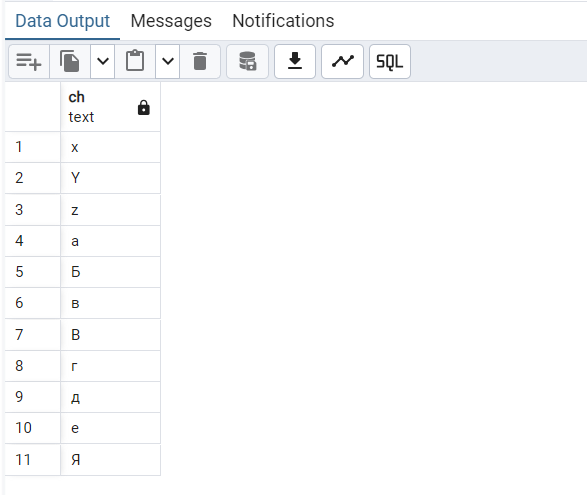
FROM unnest(ARRAY['а', 'Б', 'в', 'В', 'г', 'д', 'е', 'Я', 'z', 'Y', 'x']) AS ch

ORDER BY ch;



В этом примере массив символов интерпретируется с помощью функции unnest в строки таблицы. Как видим, сортировка выполняется в соответствии с кодами символов.

Выполним этот пример для БД с кодировкой *ru\_RU.UTF-8*:



Обратите внимание на то, что сначала идет строчная ‘в’, а потом прописная ‘В’, в том порядке, который был до выполнения сортировки, т.е. эти символы считаются одинаковыми при сортировке.

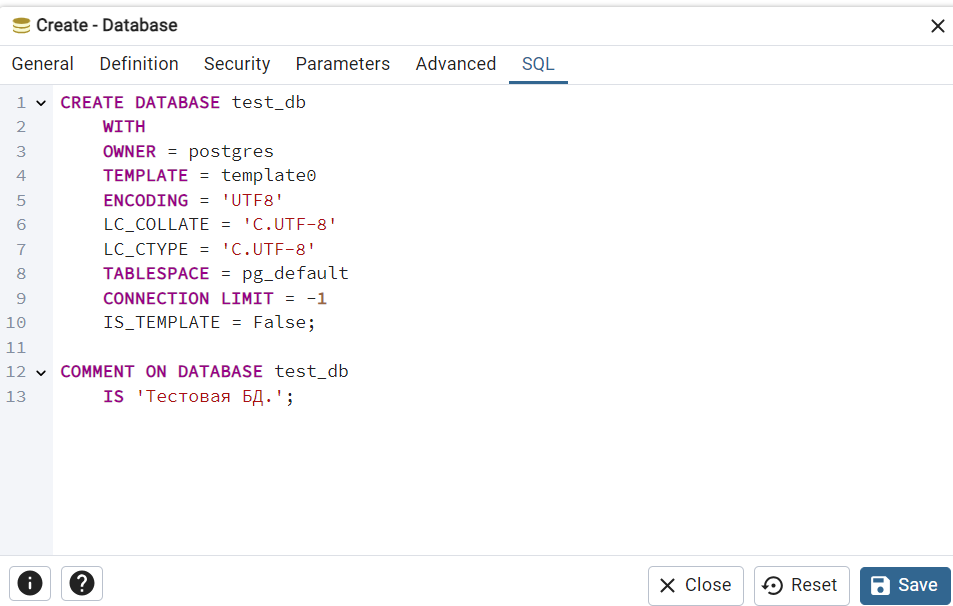
* *Connection Limit* – максимальное количество подключений к БД. Значение по умолчанию ‘*-1’* разрешает неограниченное количество подключений к БД.
* *Template?* – если установить в положение *включено* (*true*), то БД будет шаблонной.

Во вкладке *Security* устанавливаются права доступа пользователям к создаваемой БД. Этот раздел не рассматривается в данной лабораторной работе.

Вкладка *Parameters* предназначена для администрирования БД и кластера.

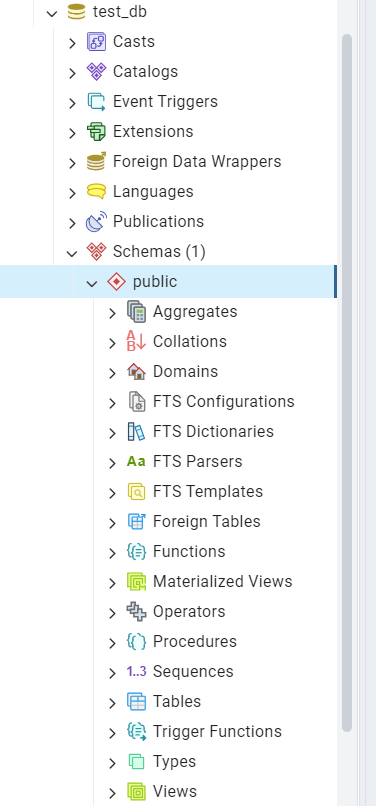
Вкладка *Advanced* предназначена для настройки браузера pgAdmin, перечисляя в поле *restriction* схемы, которые будут отображаться в дереве объектов. Эту вкладку можно пропустить.

Перейдем во вкладку *SQL*:



Здесь приведены команды на основе вашего выбора, которые будут использоваться при создании БД.

Нажмем кнопку *Save*, чтобы создалась БД.



При создании БД автоматически создается схема public.

1.2. Создание схемы

В PostgreSQL схема – это именованная коллекция объектов базы данных, включая таблицы, представления, индексы, типы данных, функции, хранимые процедуры и операторы. Схема позволяет организовывать и создавать пространства имен объектов базы данных внутри базы данных.

Чтобы получить доступ к объекту в схеме, необходимо квалифицировать объект, используя следующий синтаксис:

schema\_name.object\_name

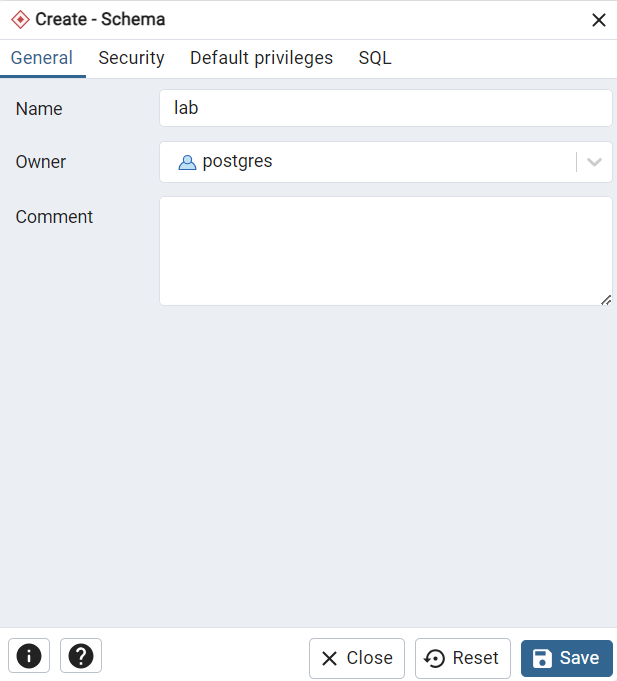
База данных может содержать одну или несколько схем. Однако схема принадлежит только одной базе данных. Кроме того, две схемы могут иметь разные объекты с одинаковым именем.

Схемы могут быть очень полезны в следующих сценариях:

* Схемы позволяют организовывать объекты базы данных, например таблицы, в логические группы, чтобы сделать их более управляемыми.
* Схемы позволяют нескольким пользователям использовать одну базу данных, не мешая друг другу.

PostgreSQL автоматически создает схему, вызываемую public для каждой новой базы данных. Какой бы объект вы ни создали, не указав имя схемы, PostgreSQL поместит его в эту public схему.

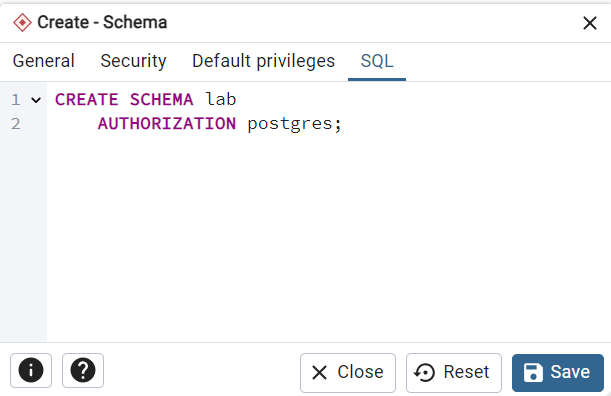
Для создания схемы щелкните правой кнопкой по узлу Schemas нужной БД и выберите команду меню *Create/Schema…*. Далее откроется диалог:



В поле *Name* необходимо ввести наименование новой схемы.

Вкладка *Security* предназначена для назначения привилегий для схемы. А вкладка *Default Privileges* – для предоставления привилегий для таблиц, последовательностей, функций и типов. Рассмотрение этих вкладок пока пропустим.

Для просмотра скрипта перейдем на вкладку *SQL*.



Создадим схему, нажав на кнопку *Save*.

1.3. Создание таблиц

Создадим таблицы group и student в схеме lab. Усложним эти таблицы добавив поля.

**Группа** (group)

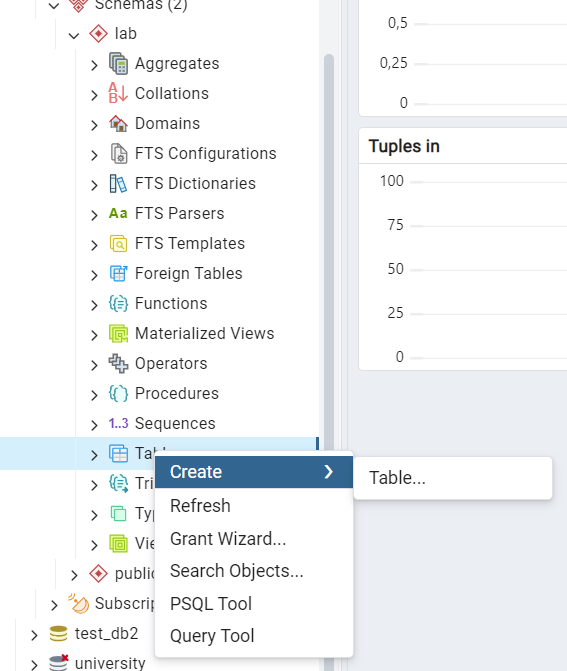
|  |  |  |
| --- | --- | --- |
| **Атрибут** | **Тип** | **Описание** |
| id\_group | serial | Суррогатный первичный ключ |
| chair | varchar(10) | Кафедра |
| num\_group | smallint | Номер группы |
| year\_start | smallint | Год создания группы |
| year\_end | smallint | Год расформирования группы (завершение обучения) |

**Студент** (student)

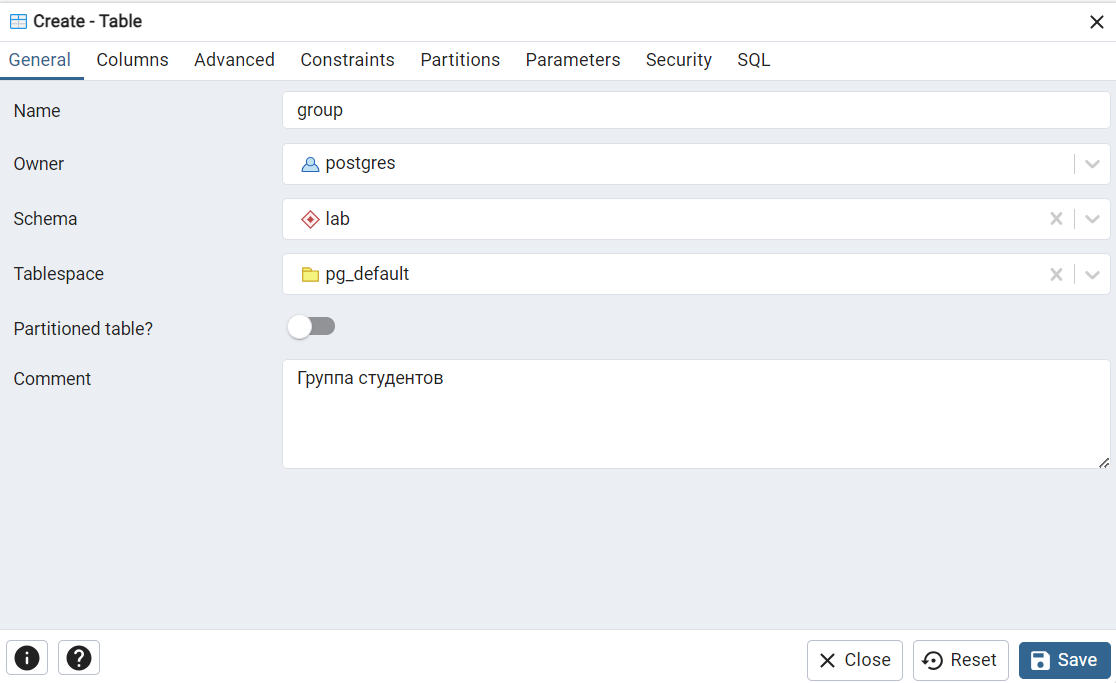
|  |  |  |
| --- | --- | --- |
| **Атрибут** | **Тип** | **Описание** |
| id\_student | serial | Суррогатный первичный ключ |
| id\_group | int | id группы, внешний ключ |
| name | varchar(50) | Фамилия, имя, отчество |
| card | char(5) | Номер студенческого билета |
| year\_start | smallint | Год поступления |
| year\_end | smallint | Год завершение обучения |

Назначение полей этих таблиц очевидна. Предполагается, что будем хранить данные за достаточно длительный период, превышающий сроки обучения. Поэтому полное название разных групп может совпадать через какой-то период. Но тройка полей chair, num\_group и year\_start должна быть уникальна. Уникальна должна быть пара card, year\_start.

Приступим к созданию таблиц. Щелкнем правой клавишей по узлу *Tables* в схеме *lab* и выберем пункт меню *Create/Table…*

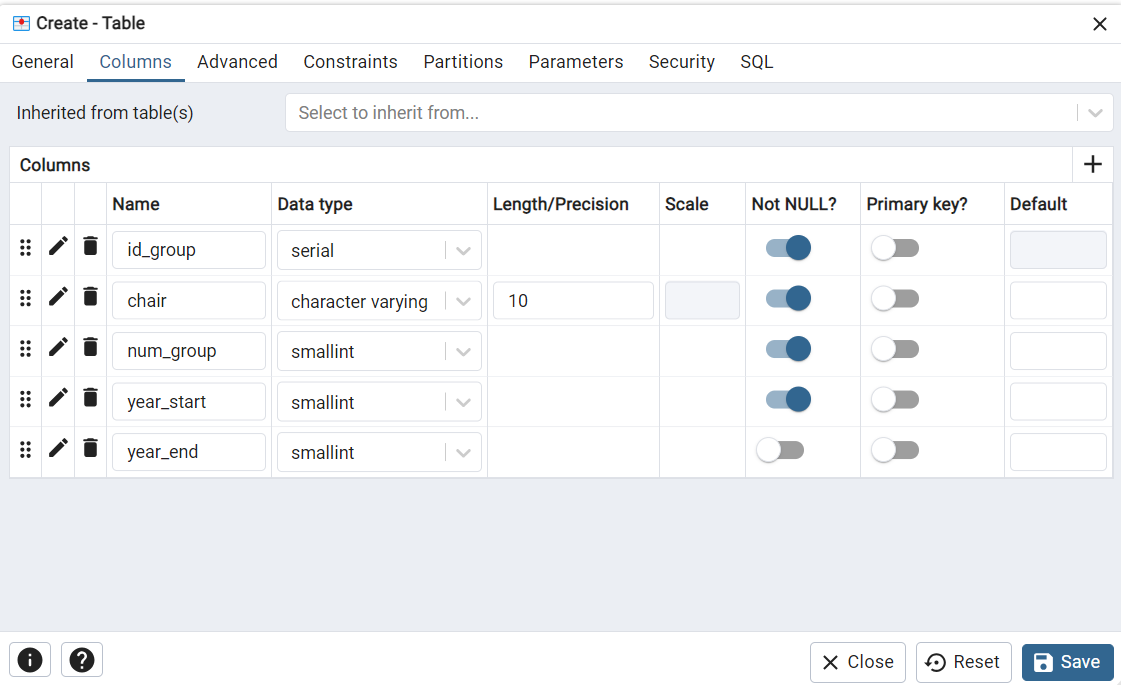


Откроется диалог создания таблицы:



На вкладке General назначение полей не должно вызвать затруднений, кроме переключателя *Partitioned Table*. Его используют для секционирования (разбиения) больших таблиц. Эту тему мы опустим, так что оставим выключатель не активным.

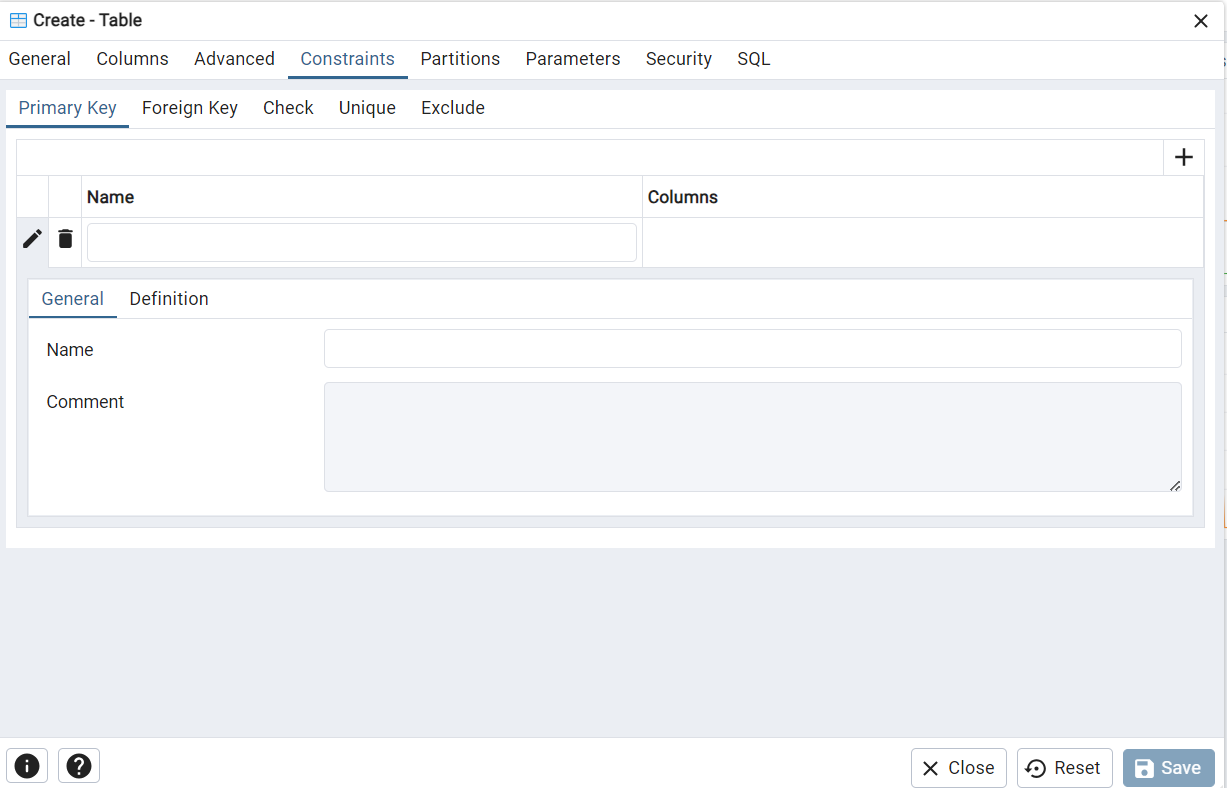
Продолжим, щелкнув по вкладке *Columns*. В этой вкладке введем поля и их типы, отметим так же какие поля не могут иметь значения NULL. Значение NULL для поле year\_end означает, что группа обучается.



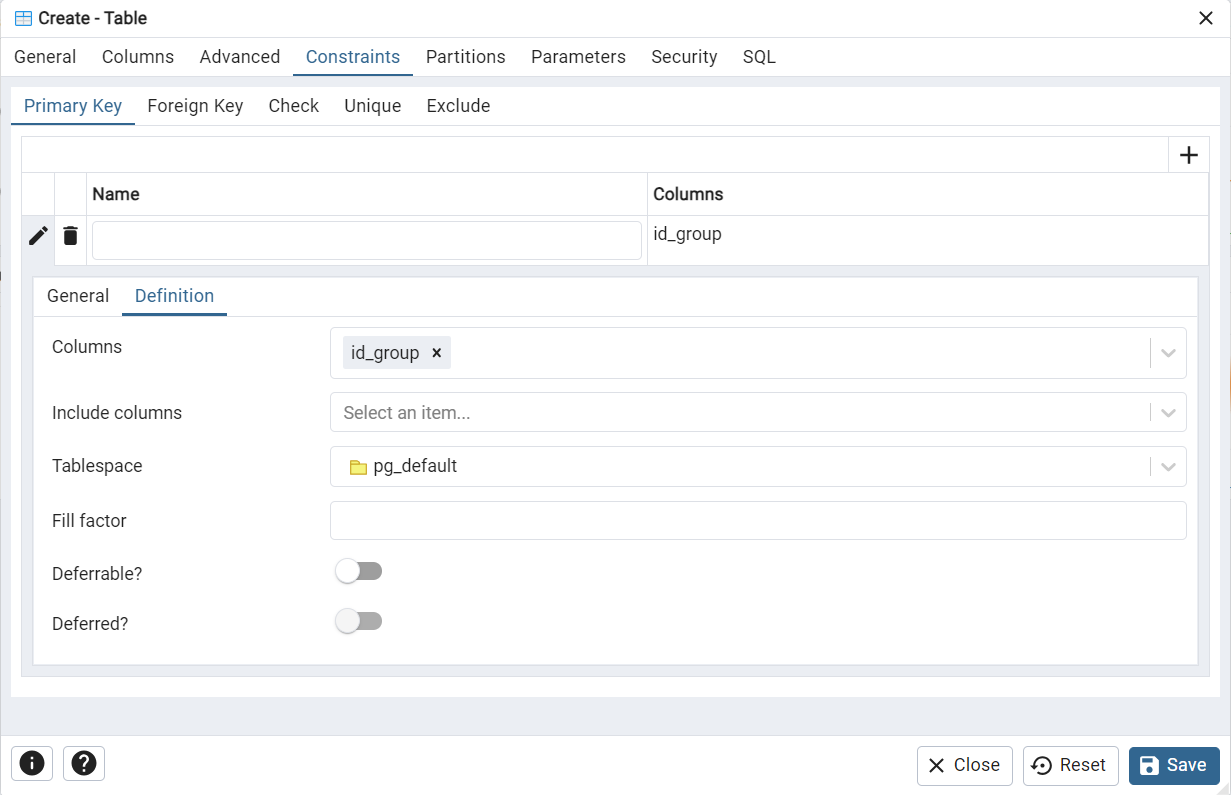
Здесь можно отметить поля, которые образуют первичный ключ (id\_group), но мы сделаем это в другом месте.

Пропустим вкладку Advanced, на начальном этапе ознакомления с PostgreSQL она не потребуется, и перейдем к вкладке *Constraints* (ограничения):

Чтобы добавить первичный ключ, нажмем на «плюс» на вкладке *Primary Key*:



В поле Name можно ввести название первичного, если этого не сделать, название будет сформировано автоматически – <таблица>\_pkey, что вполне разумно. Так что перейдем к вкладке *Definition* и зададим поля первичного ключа:



В *Columns* следует указать поля таблицы, образующие первичный ключ, последовательно выбирая их из выпадающего списка. В нашем случае он состоит из одного поля. Остальные поля и переключатели оставим без изменения.

Таблица group не имеет внешних ключей, переходим к ограничениям *Check*, *Unique[[3]](#footnote-3)*.

В PostgreSQL *Check* ограничение гарантирует, что значения в столбце или группе столбцов соответствуют определенному условию.

Ограничение проверки позволяет обеспечить соблюдение правил целостности данных на уровне базы данных. Проверочное ограничение использует логическое выражение для оценки значений, гарантируя, что в таблицу вставляются или обновляются только допустимые данные.

Введем следующие ограничения в вкладку *Check*:

* Название кафедры не может состоять из пробельных символов. Это условие запишем таким образом:

trim(chair) <> ''

Здесь функция trim удаляет концевые пробелы и, если результат не равен пустой строке, то введенное название считаем допустимым.

* Номер группы должен быть больше нуля:

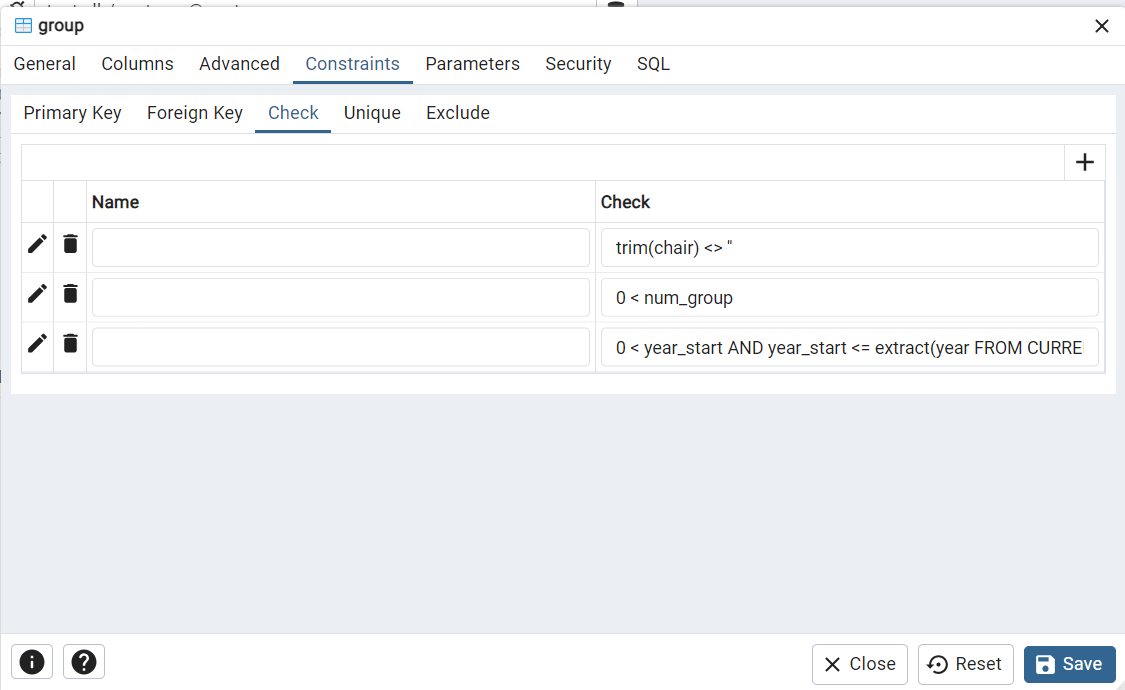
0 < num\_group

* Год создания группы должен быть больше нуля и не больше текущего года:

0 < year\_start AND year\_start <= extract(year FROM CURRENT\_DATE)

Функция extract возвращает год текущей даты CURRENT\_DATE.

В итоге вкладка *Check* примет вид:



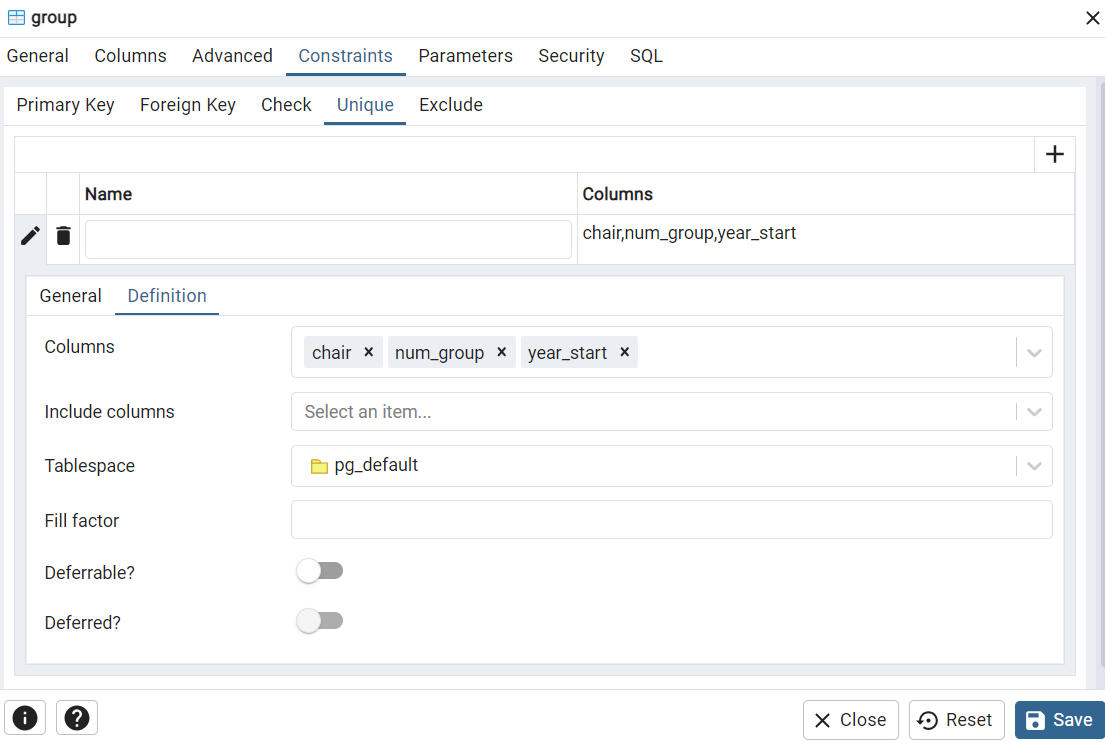
Перейдем к вкладке *Unique*. Иногда требуется убедиться, что значения, хранящиеся в столбце или группе столбцов, являются уникальными во всей таблице.

PostgreSQL предоставляет вам ограничение UNIQUE, которое корректно поддерживает уникальность данных.

Когда ограничение UNIQUE установлено, каждый раз, когда вы вставляете новую строку, он проверяет, есть ли уже значение в таблице. Он отклоняет изменение и выдает ошибку, если значение уже существует. Тот же процесс выполняется при изменении (UPDATE) существующих данных.

При добавлении ограничения UNIQUE к столбцу или группе столбцов PostgreSQL автоматически создаст уникальный индекс для столбца или группы столбцов.

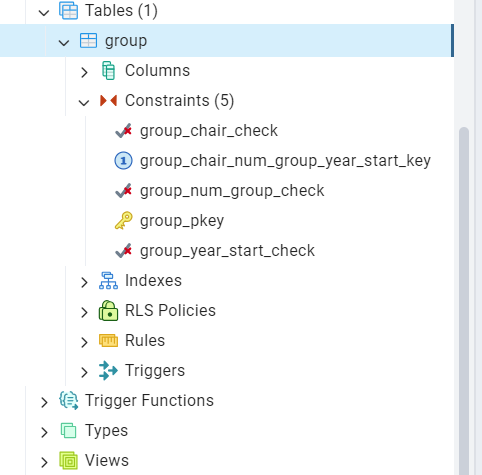
Как отмечалось выше, поля chair, num\_group и year\_start однозначно определяют группу, т.е. не должно быть двух и более строк, имеющих одинаковые значения этих полей. Перечислим chair, num\_group и year\_start в поле *Columns*:



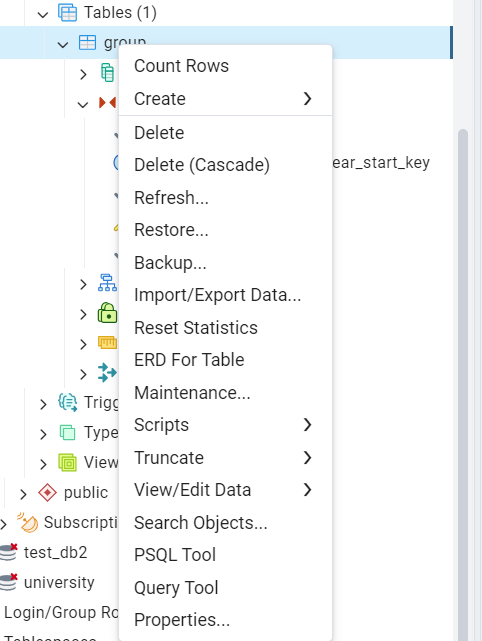
Все остальные параметры оставим без изменения. Как обычно, скрипты, запускаемые для создания таблицы, отображены в вкладке *SQL*.

Создадим таблицу, нажав кнопку *Save*.

В дереве объектов отобразится таблица и введенные ограничения:

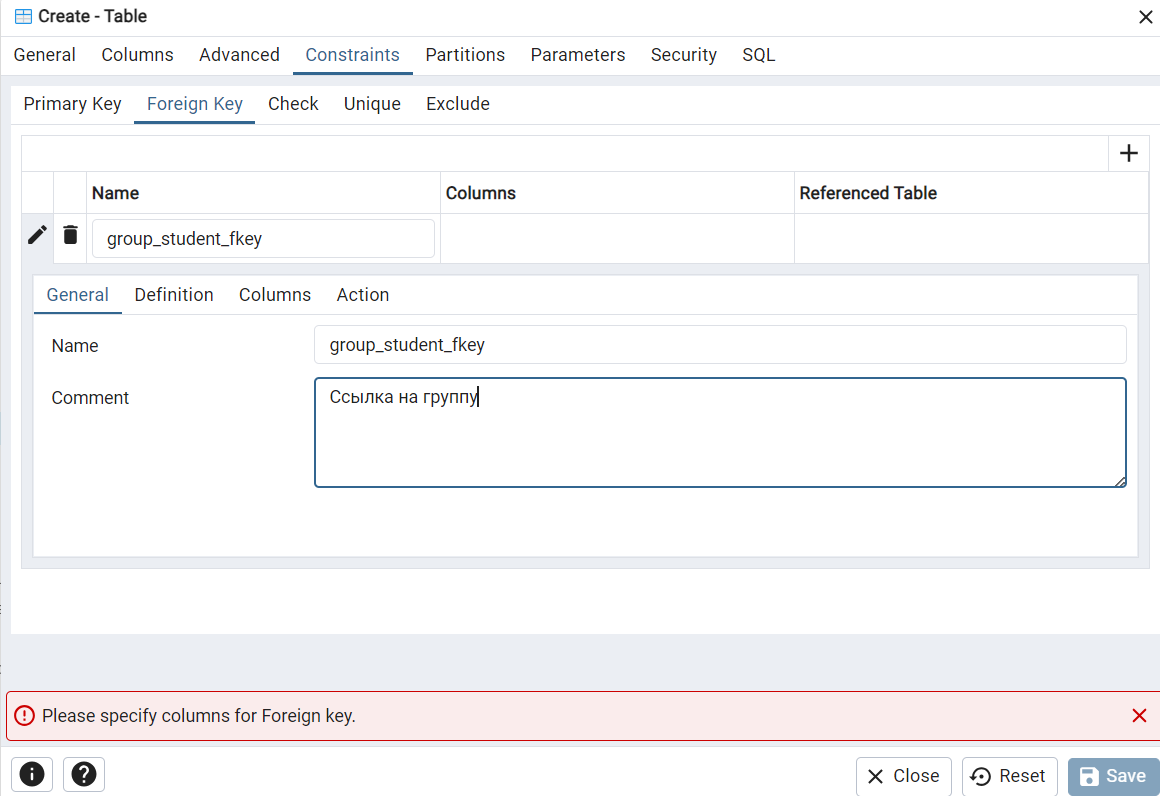


Если потребовалось изменить атрибуты в таблице, то нажмите правую клавишу и выберите пункт меню *Properties*:

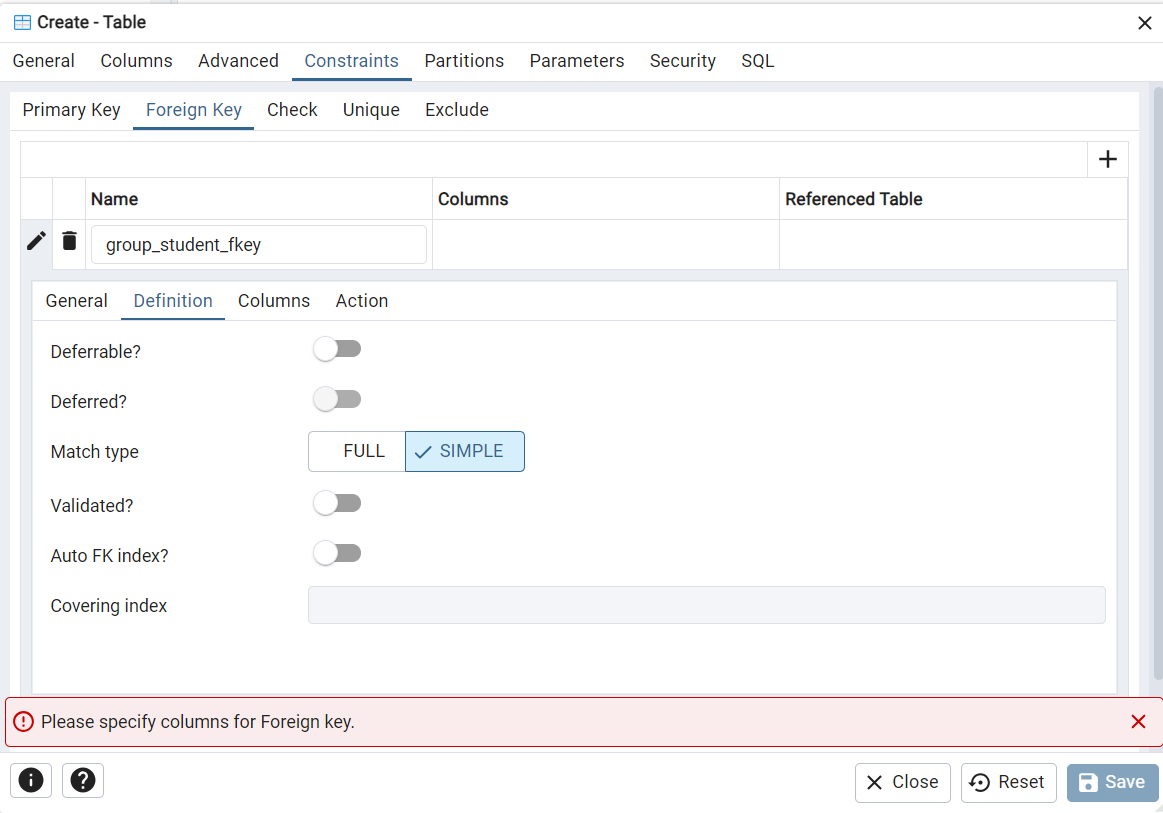


Откроется диалог, что и при создании таблицы, в котором можно ввести необходимые изменения.

Прейдем к созданию таблицы student. Создание таблицы student аналогично созданию таблицы group, кроме создания ограничения внешнего ключа. Так что после ввода полей, задания первичного ключа и т.д., перейдем во вкладку *Foreign Key*. Во вкладке *General* можно задать необязательное имя внешнего ключа:



Перейдем на следующую вкладку – *Definition*:

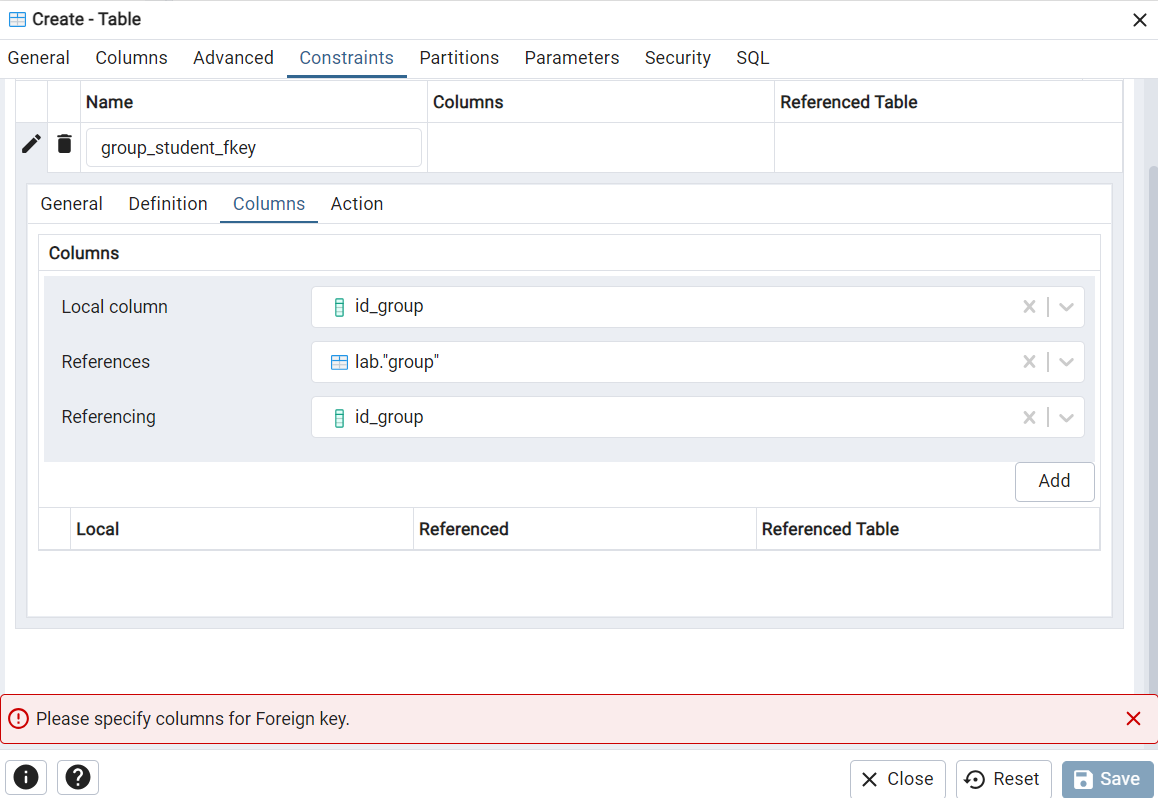


Переключатель *Match type* необходим для внешнего ключа, состоящего из нескольких столбцов (составного), чтобы указать тип соответствия:

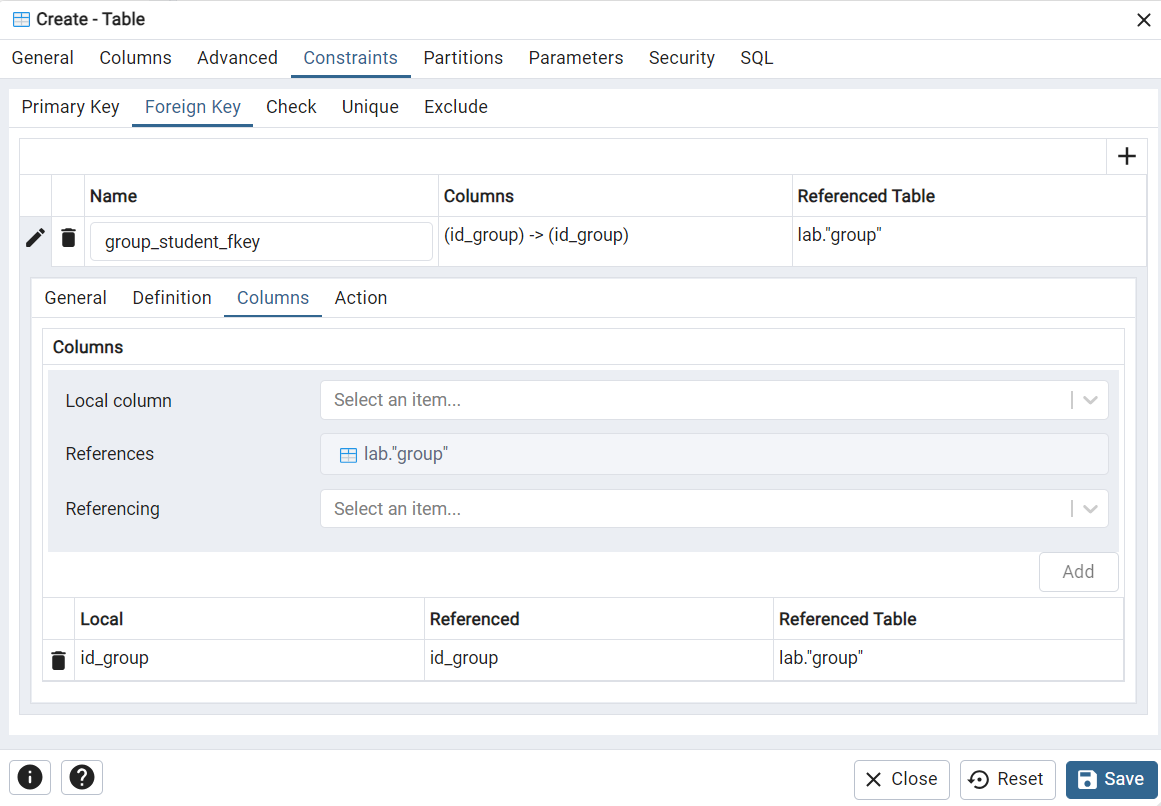
* Выберите *Full*, чтобы указать, что ни один из столбцов составного внешнего ключа не может содержать NULL, кроме случая, когда все внешние столбцы NULL. В этом случае строка может не иметь соответствия во внешней таблице.
* Выберите *Simple*, чтобы указать, любой из столбцов внешнего ключа может содержать NULL, при этом строка с NULL в одном из таких столбцов может не иметь соответствия во внешней таблице.

В нашем случае этот переключатель не на что не влияет. Оставим остальные переключатели без изменения.

Рассмотрим вкладку *Columns*.

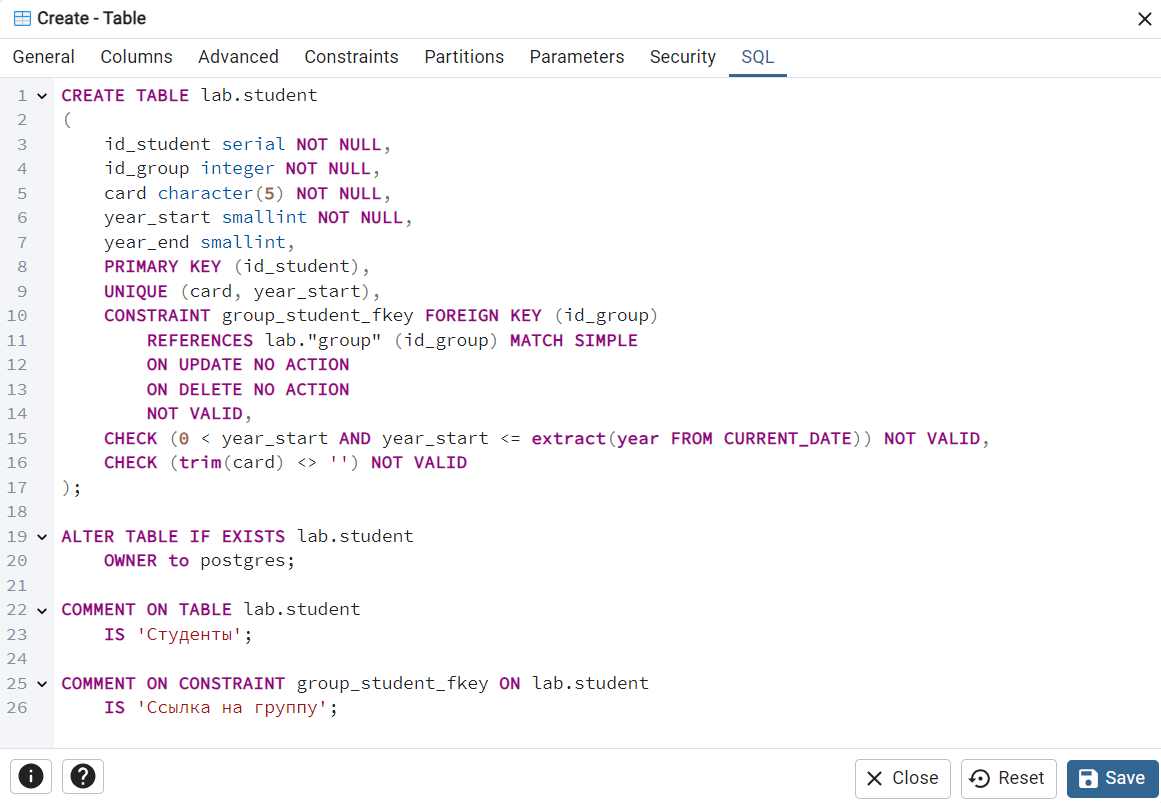


В *Local column* выбираем поле внешнего ключа текущей таблицы student. Далее выбираем из ниспадающих списков таблицу group и поле, соответствующее внешнему ключу – id\_group. Нажмем на кнопку *Add*,чтобы определить внешний ключ:



Следующая вкладка *Action* рассмотрена в предыдущей лабораторной работе.

Нажмем на вкладку *SQL*, чтобы посмотреть на результат своей работы:



Сохраним, чтобы создать таблицу student.

## Контрольные вопросы

1. Для чего нужны БД template0, template1?
2. Кодировка символов, что это?
3. Что такое первичный ключ?
4. Что такое внешний ключ?
5. Какие существуют способы поддержания ссылочной целостности?
6. Какие ограничения целостности знаете?

## 3. Практическое задание

1. На основе разработанной ER–модели, создать БД на сервере PostgreSQL.
2. Добавить в таблицы необходимые ограничения *NOT NULL*, *CHECK* и *UNIQUE* и обосновать их необходимость.

## 4. Список использованных источников

1. Postgres Pro Standard Документация.

<https://postgrespro.ru/docs/postgrespro>

1. pgAdmin Documentation.

<https://www.pgadmin.org/docs/>

1. При установке PostgreSQL создается суперпользователь postgres. [↑](#footnote-ref-1)
2. Для создания БД с требуемой кодировкой следует воспользоваться командой CREATE DATABASE. [↑](#footnote-ref-2)
3. Ограничение Exclude рассматривать не будем. [↑](#footnote-ref-3)