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 № 4** **«****Введение в PostgreSQL. ER–модель»**

Авторы:

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

Фомин М.М.

Ланцберг А.В.

Москва, 2024

## Основы языка SQL

1.1. Первичный ключ

Первичный ключ — это столбец или группа столбцов, используемых для уникальной идентификации строки в таблице. Столбец, который участвует в первичном ключе, называется столбцом первичного ключа. Поля, входящие в первичный ключ, не могут быть NULL. Таблица может не иметь первичный ключ. Кроме того, она не может иметь более одного первичного ключа.

Первичный ключ важен, потому что он гарантирует отсутствие дубликатов строк в таблице, а также позволяет эффективно выполнять запросы и индексировать таблицу.

Хорошей практикой является добавление первичного ключа к каждой таблице. Когда вы добавляете первичный ключ к таблице, PostgreSQL создает уникальный индекс B-дерева для столбца или группы столбцов, используемых для определения первичного ключа.

Если вы явно не укажете имя ограничения первичного ключа, PostgreSQL присвоит ограничению первичного ключа имя по умолчанию. По умолчанию в качестве имени первичного ключа PostgreSQL использует формат table-name\_pkey.

1.2. Внешний ключ

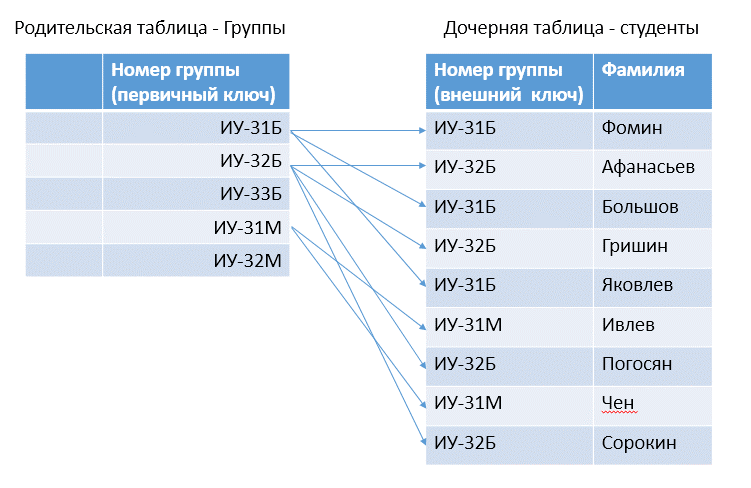
В PostgreSQL внешний ключ — это столбец или группа столбцов в таблице, которые уникально идентифицируют строку в другой таблице. Внешний ключ устанавливает связь между данными в двух таблицах, ссылаясь на первичный ключ указанной таблицы.

Таблица, содержащая внешний ключ, называется дочерней таблицей. И наоборот, таблица, на которую ссылается внешний ключ, называется родительской таблицей.

Основная цель внешних ключей — поддерживать ссылочную целостность в реляционной базе данных, гарантируя, что связи между родительскими и дочерними таблицами являются действительными. Например, внешний ключ предотвращает вставку значений, не имеющих соответствующих значений в указанной таблице. Кроме того, внешний ключ поддерживает согласованность, автоматически обновляя или удаляя связанные строки в дочерней таблице при возникновении изменений в родительской таблице.

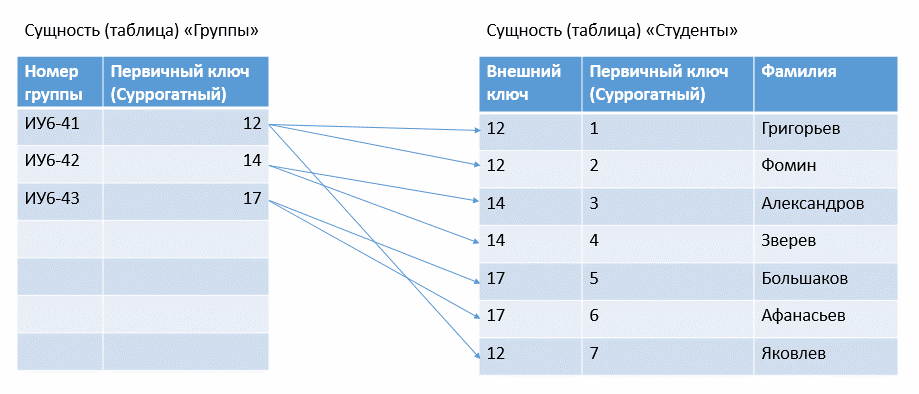
Таблица может иметь несколько внешних ключей в зависимости от ее связей с другими таблицами.

Рассмотрим пример. В таблицах на рисунке присутствуют поле «Номер группы» в таблице «Группа» — это первичный ключ, а в таблице «Студент» поле «Номер группы» — это внешний ключ, определяющий группу, в которой учится студент.



Зная экземпляр группы и, следовательно, значение первичного ключа – «Номер группы» (НГ), можно по значению внешнего ключа равному НГ выбрать всех студентов этой группы и, наоборот, по значению внешнего ключа (НГ конкретного студента) можно найти нужный экземпляр сущности «Группа». Так работают связи типа «один ко многим» для таблиц (сущностей) в реляционной СУБД.

Но ведь номер группы меняется каждый семестр и, следовательно, связывать таблицы по этому атрибуту нежелательно. При проектировании БД полезно сохранять значения первичных ключей неизменными на протяжении жизненного цикла БД[[1]](#footnote-1). Выходом является использование в качестве первичного ключа суррогатного ключа. Суррогатный ключ — это целочисленный атрибут, значение которого генерируется автоматически при добавлении экземпляра сущности (записи в таблицу). В PostgreSQL для этих целей предназначены типы данных из семейства serial и NOTIFY. При использовании суррогатных ключей связи между таблицами получаются как на рисунке.



Таким образом, номер группы меняется каждый семестр, но это нисколько не затрагивает связи в базе данных, поскольку суррогатный первичный ключ группы не меняется на протяжении всего жизненного цикла группы с момента ее создания и до перемещения данных о группе в архив. Этот суррогатный первичный номер группы уникален во всем вузе и на все времена.

## Проектирование схемы базы данных

2.1. Модель «сущность – связь»

Модель «сущность–связь» или ER–модель представляет собой схему, составными элементами которой являются:

**Сущность** — это реальный, либо воображаемый объект, информацию о котором необходимо хранить в базе данных. На диаграмме ER–модели сущность изображается в виде прямоугольника, содержащего имя сущности.

**Связь** — отображаемая графически на диаграмме ассоциация между двумя (чаще всего) сущностями, или между одной и той же сущностью (рекурсивная связь).

В системе сущность представлена в виде экземпляров. Например, экземпляры сущности «Аэропорт» ― аэропорты «Домодедово», «Пулково», «Воронеж».

У сущностей есть атрибуты — характеристики, которые их описывают. Например, атрибутами сущности «Аэропорт» будут код, адрес, номер телефона. Атрибуты есть у каждого экземпляра сущности, но у них разные значения. У аэропортов «Домодедово» и «Воронеж» есть одинаковый атрибут «Адрес», но у каждого из них разное значение этого атрибута.

В ER–модели есть три типа связей:

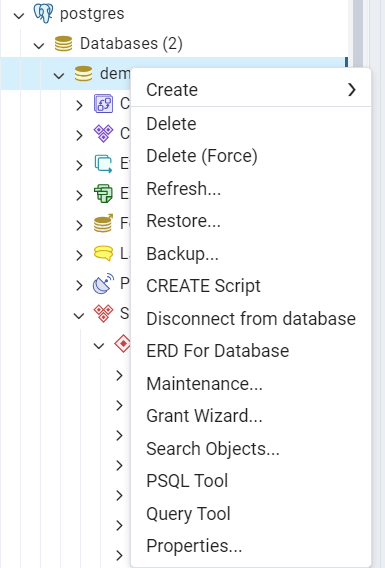
● «Один-к-одному» — один экземпляр сущности связан только с одним экземпляром другой сущности. Например, пассажир рейса и его место в самолете.

● «Один-ко-многим» — один экземпляр сущности связан со множеством экземпляров другой сущности. Например, у одного пассажира может быть несколько единиц багажа, при этом каждая единица багажа может быть связана только с одним пассажиром.

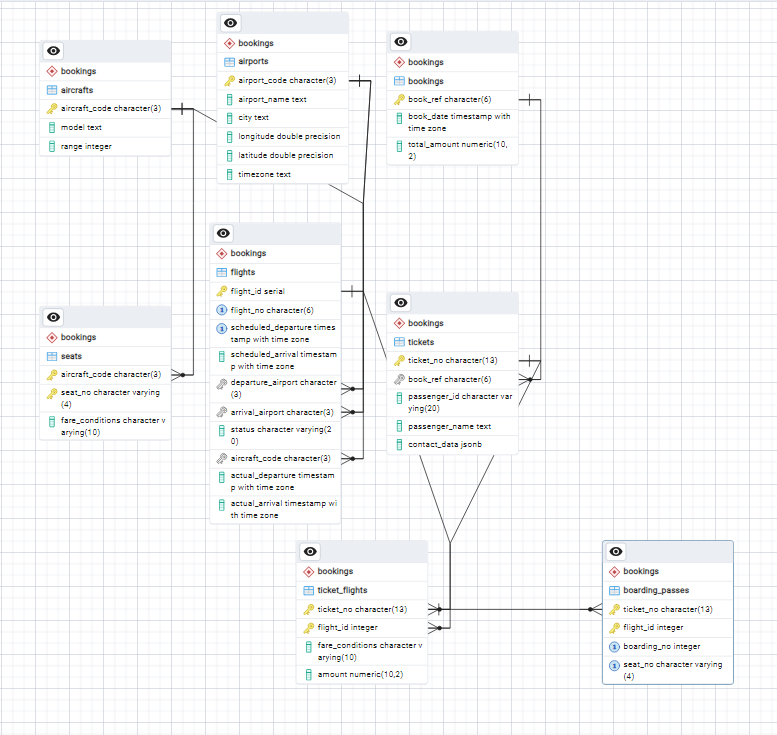
● «Многие-ко-многим» — множество экземпляров одной сущности связаны со множеством экземпляров другой сущности. Например, аэропорт обслуживает несколько авиакомпаний. При этом каждая авиакомпания может обслуживаться в нескольких аэропортах.

pgAdmin может строить ER–модели для БД. Сформируем с помощью pgAdmin ER–модель для БД demo.

Нажмем правой клавишей на узел demo и выберем команду меню *ERD For Database*.



После этого отобразится схема:



Прямоугольники сущностей и связи можно перемещать, делая схему более читаемой.

Схема БД demo использует нотацию Мартина (Crow's Foot Notation). Связи изображаются линиями, соединяющими сущности, вид линии в месте соединения с сущностью определяет кардинальность связи:

|  |  |
| --- | --- |
| **Обозначение** | **Кардинальность** |
|  | один |
|  | один и только один |
|  | ноль или один |
|  | много |
|  | ноль или много |
|  | один или много |

2.2. Построение ER–модели БД

Для проектирования БД существует несколько средств разработки структуры БД: ERwin, MS Visio, Lucidchart и др. Но мы будем использовать инструмент, предоставляемый pgAdmin.

Будем разрабатывать упрощенную БД ВУЗ (university), которая должна обеспечить составления расписания занятий для групп. Очевидно, что в нее должны входить сущности «студент» и «группа». Отношение между двумя этими сущностями один-ко-многим, так как группа состоит из нескольких студентов, и каждый студент принадлежит только одной группе. Этот факт отмечается наличием в сущности «студент» внешнего ключа (id\_group), ссылающегося на строку в сущности «группа».

**Группа** (group)

|  |  |  |
| --- | --- | --- |
| **Атрибут** | **Тип** | **Описание** |
| id\_group | serial | Суррогатный первичный ключ |
| name | varchar(10) | Группа |

**Студент** (student)

|  |  |  |
| --- | --- | --- |
| **Атрибут** | **Тип** | **Описание** |
| id\_student | serial | Суррогатный первичный ключ |
| id\_group | int | id группы, внешний ключ |
| name | varchar(50) | Фамилия, имя, отчество |
| … | ... | ... |

Сущность «аудитория»**[[2]](#footnote-2)** тоже очевидна.

**Аудитория** (auditorium)

|  |  |  |
| --- | --- | --- |
| **Атрибут** | **Тип** | **Описание** |
| id\_auditorium | serial | Суррогатный первичный ключ |
| room\_no | varchar(10) | Номер помещения |
| … | … | … |

Перейдем к сущности «расписание». В каждой аудитории в разное время могут проходить разные занятия, с другой стороны, строка сущности «расписание» однозначно ссылается на сущность «аудитория». Из чего можно заключить, что связь между сущностями «аудитория» и «расписание» один-ко-многим.

Как отмечалось выше, строка сущности «расписание» должна однозначно определяться для каждой сущности «аудитории», т.е. в данный момент времени в аудитории не может проходить более одного занятия. Учитывая специфику ВУЗ’а, однозначность достигается следующими атрибутами: год, семестр, день недели, время начала занятий, продолжительность или время конца занятий и аудитория.

В «расписание» надлежит добавить атрибуты: тип занятия (лекции, семинар и т.п.), преподаватель и т.д., но на данный момент это для нас неважно и ограничимся атрибутом – тип занятия.

**Расписание** (schedule)

|  |  |  |
| --- | --- | --- |
| **Атрибут** | **Тип** | **Описание** |
| id\_schedule | serial | Суррогатный первичный ключ |
| id\_auditorium | int | id аудитории, внешний ключ |
| year | smallint | Год |
| semester | smallint | 0 – осенний семестр, 1 – весенний |
| day\_week | smallint | День недели |
| time\_start | time | Время начала занятий |
| time\_end | time | Время конца занятий |
| type\_lesson | smallint | 0 – лекция, 1 – семинар и т.п. |
| … | … | … |

Нам осталось связать сущности «группа» и «расписание». С одной стороны то, что на занятиях может присутствовать несколько групп, в терминах БД сущности «расписание» соответствует несколько сущностей «группа». С другой, каждая группа в течения дня посещает несколько занятий, т.е. сущности «группа» соответствует несколько сущностей «расписание». Отсюда следует, что связь между группой и расписанием многие-ко-многим.

Чтобы создать связи многие-ко-многим, необходимо создать новую таблицу для подключения двух других. Эта новая таблица называется ассоциативной таблицей. Для этих связей необходимо просто соединить соответствующие поля с помощью строки.

В нашем случае, такой ассоциативной таблицей будет таблица group\_schedule. Одна строка посредством внешних ключей соединяет «группу» и «расписание».

**Группа-расписание** (group\_schedule)

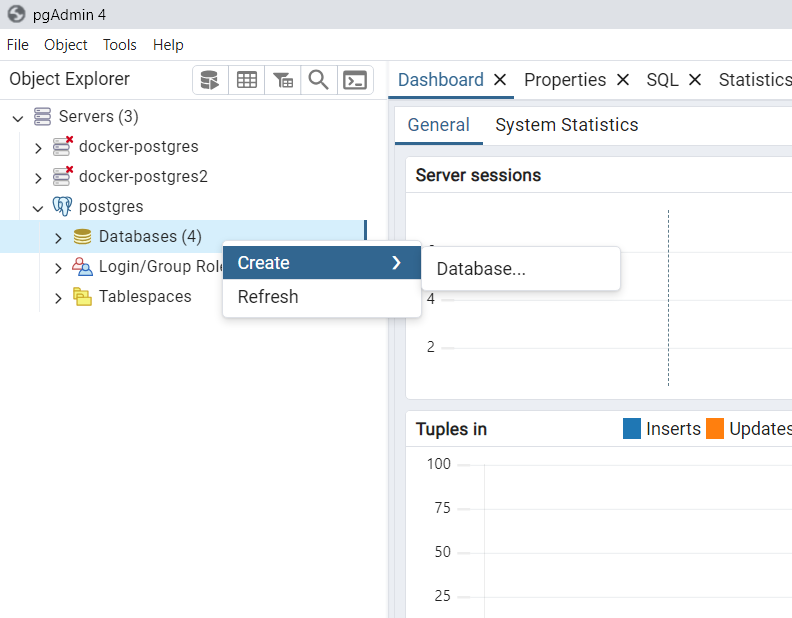
|  |  |  |
| --- | --- | --- |
| **Атрибут** | **Тип** | **Описание** |
| id\_group | int | id группы, внешний ключ |
| id\_schedule | int | id расписания, внешний ключ |

Таким образом, с помощью ассоциативной сущности связь многие-ко-многим мы представили двумя связями один-ко-многим между сущностями «группа» - «группа-расписание» и «расписание» - «группа-расписание».

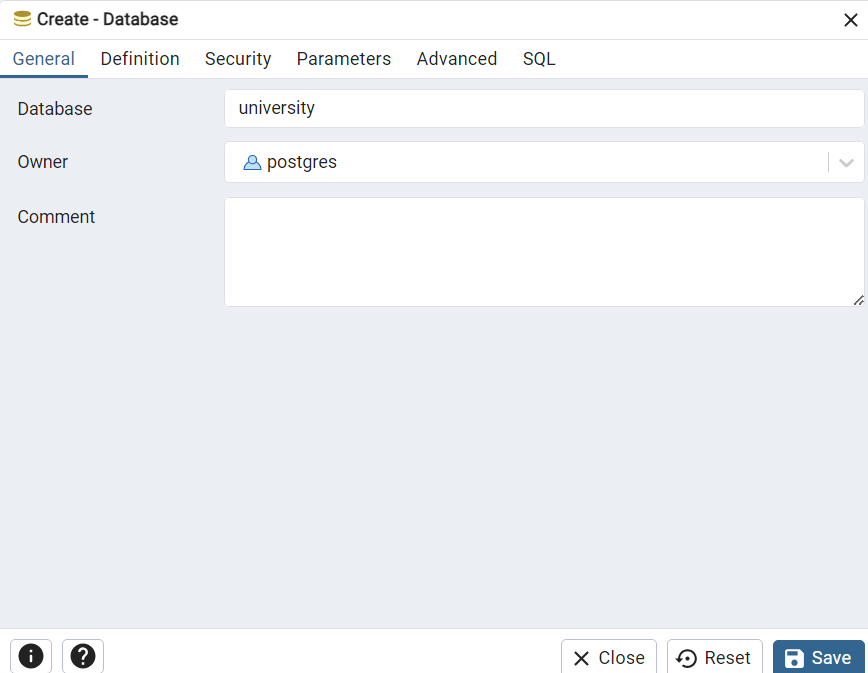
В качестве первичного ключа для сущности «группа-расписание» целесообразно использовать пару (id\_group, id\_schedule).

Займемся построением ER–модели. Запустим pgAdmin. Перед тем, как строить ER–модель, необходимо создать БД university.

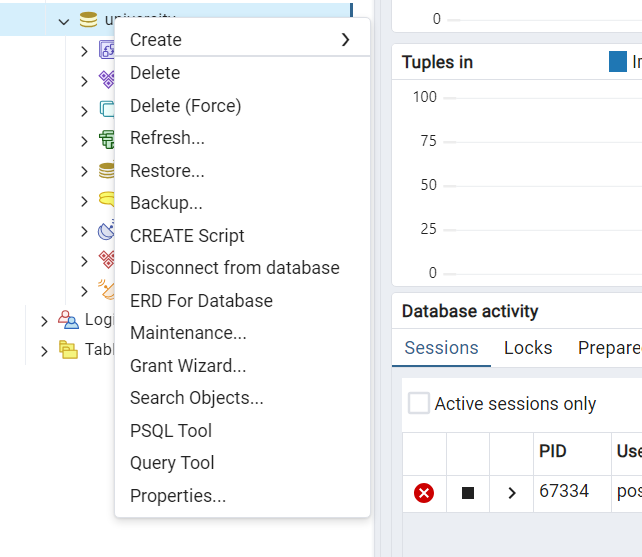
Щелкните правой кнопкой по узлу Databeses и выберите пункт меню *Create/ Database...*.



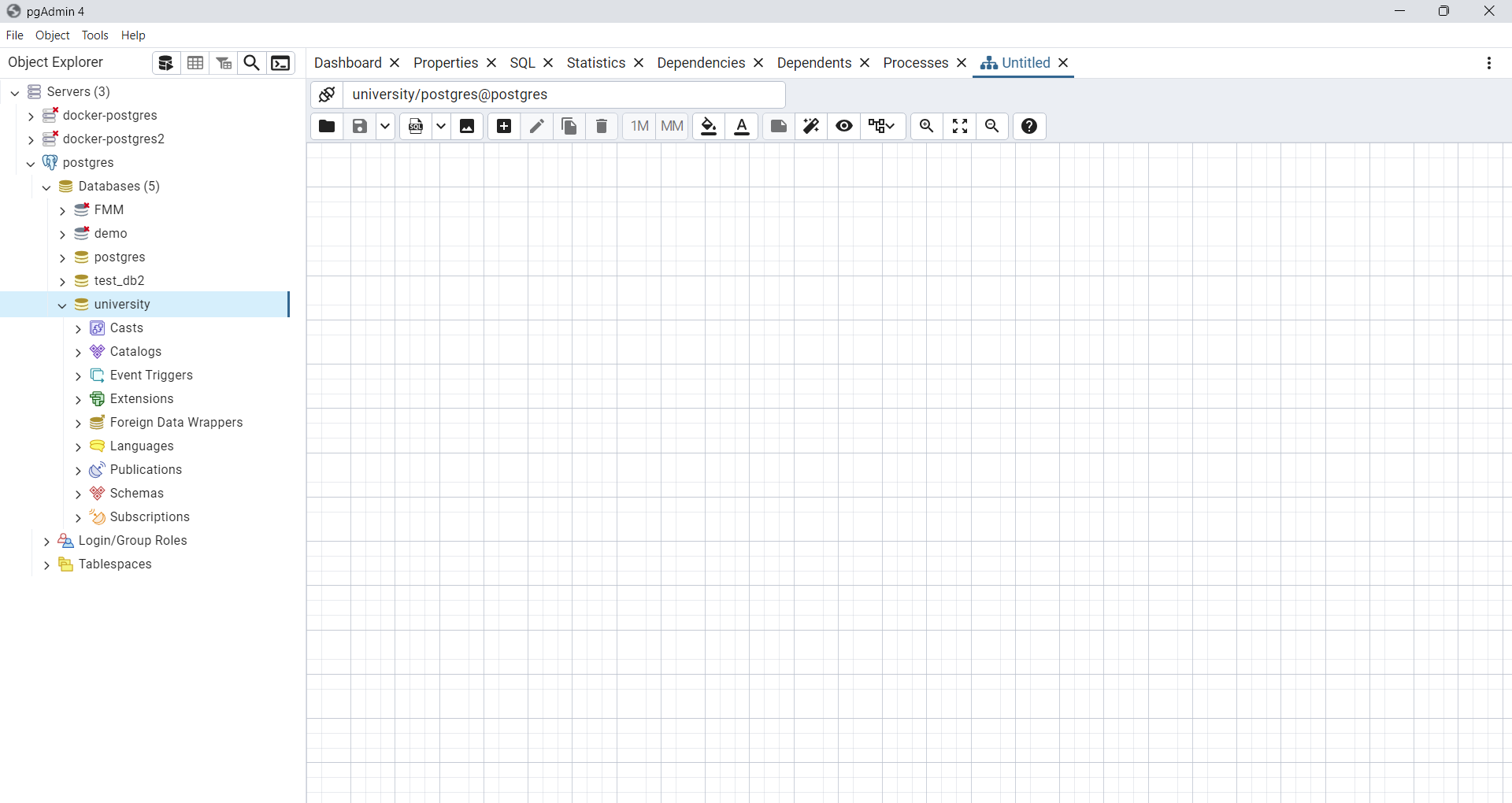
Появится диалоговое окно для ввода подробной информации о новой базе данных.



Введем название новой БД и сохраним. Щелкните правой кнопкой по узлу university и выберем команду меню *ERD For Database.*



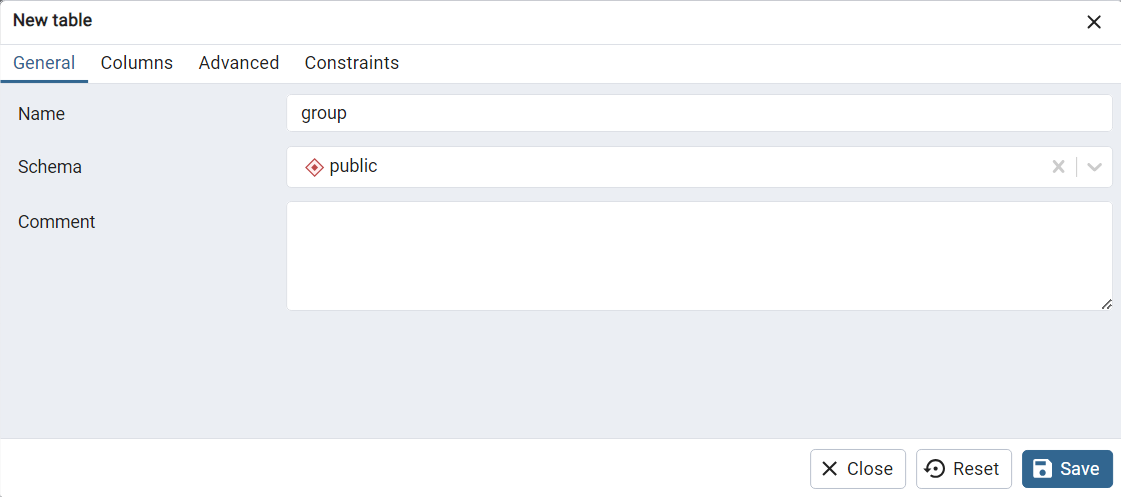
Откроется окно для проектирования БД.



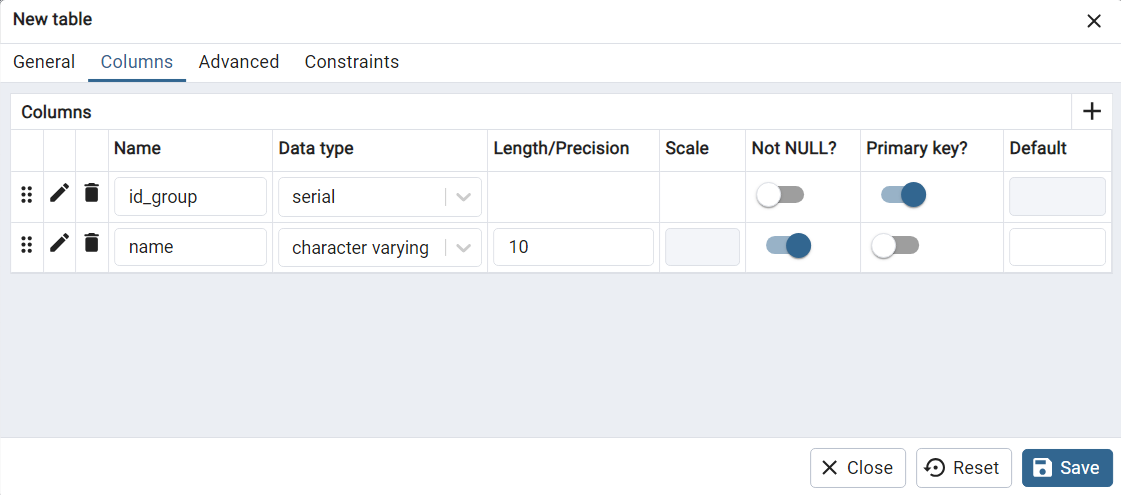
Панель инструментов ERD *Tool* использует контекстно-зависимые значки, которые обеспечивают ярлыки для часто выполняемых задач. Опция включена для выделенного значка и отключена для серого значка.

Панель инструментов ERD

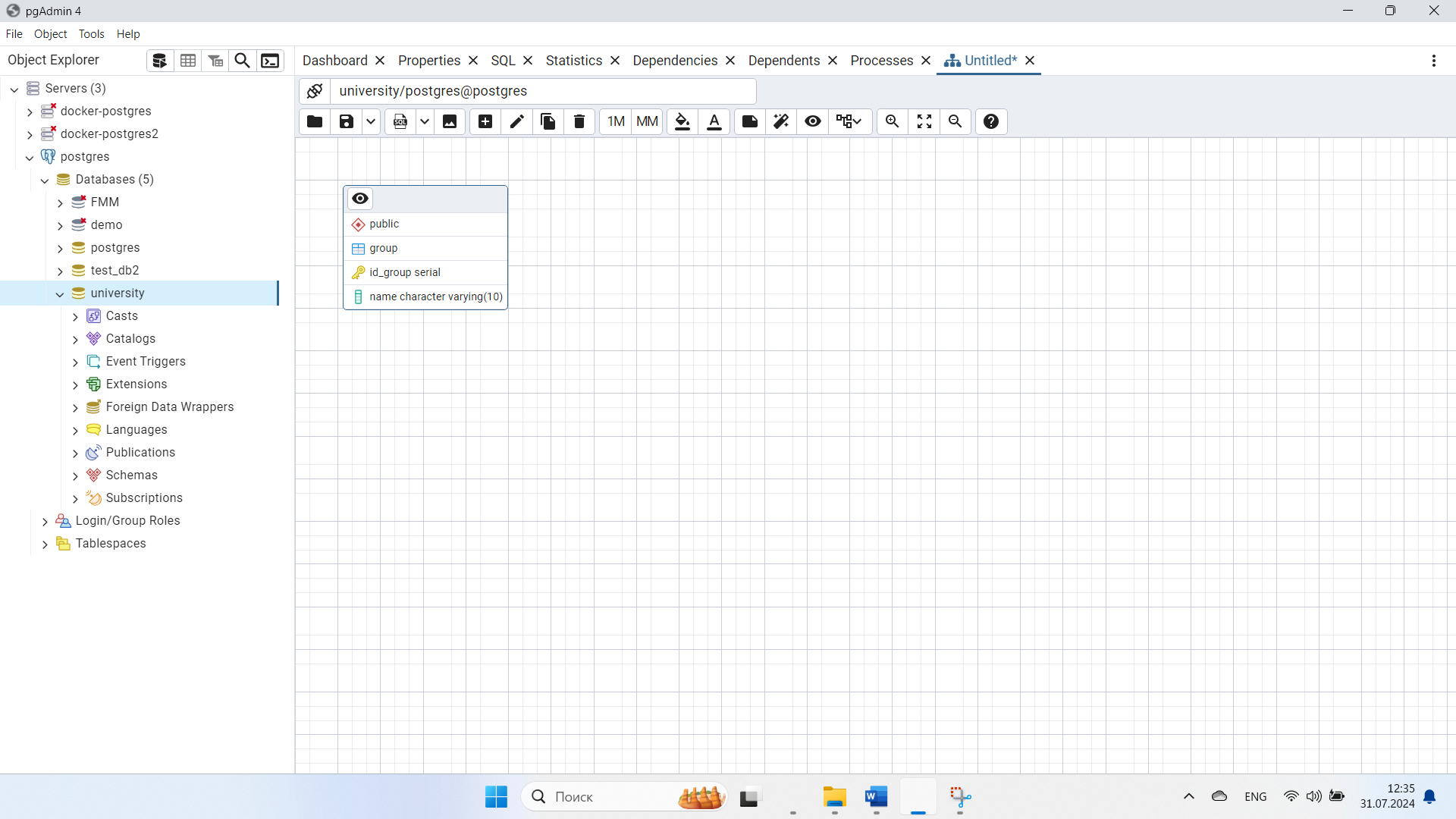
Нажмем на значок , чтобы приступить к созданию таблицы. Откроется диалоговое окно таблицы:



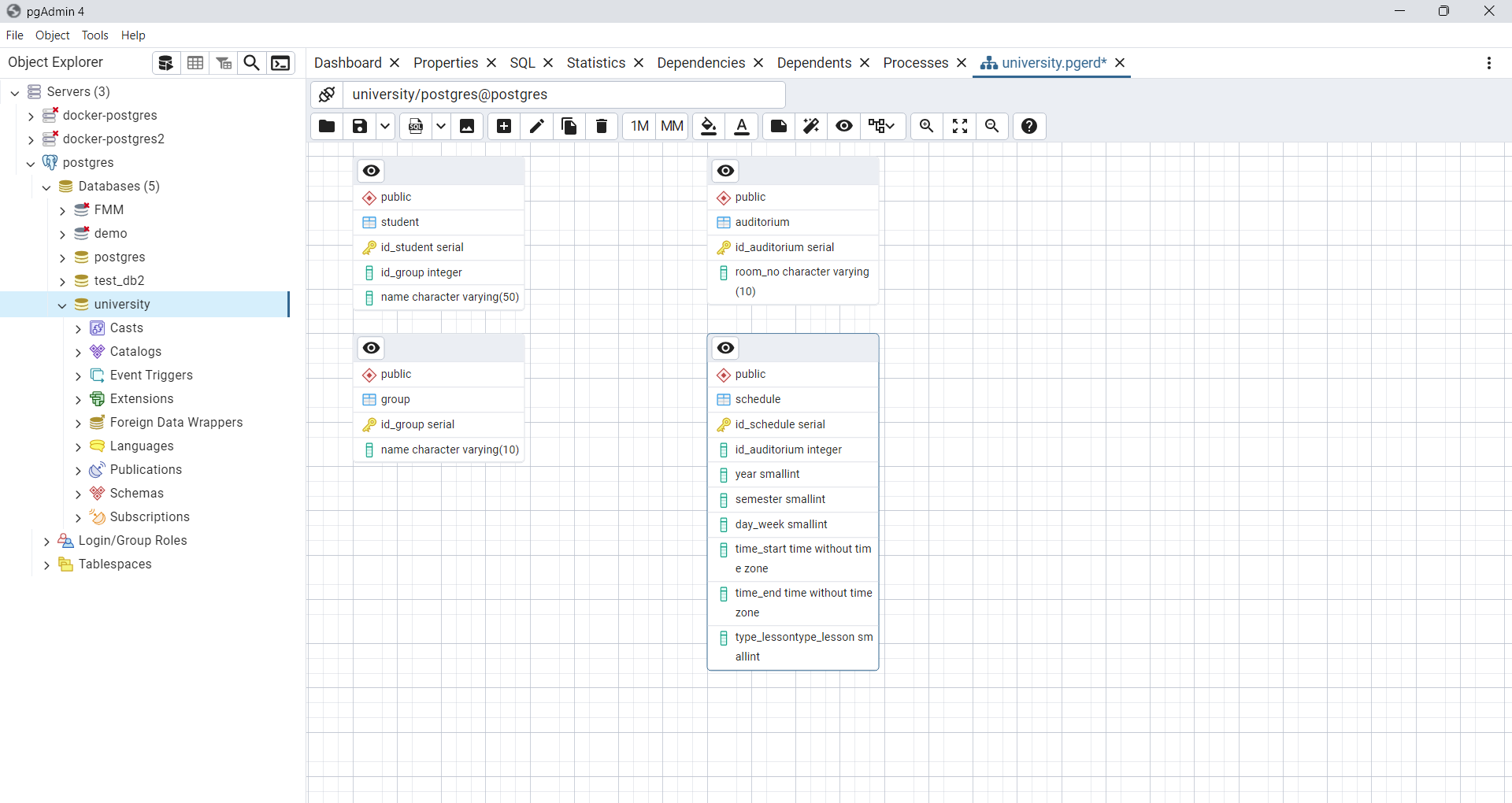
Введем название таблицы и перейдем к вводу атрибутов (полей), выбрав вкладку Columns:



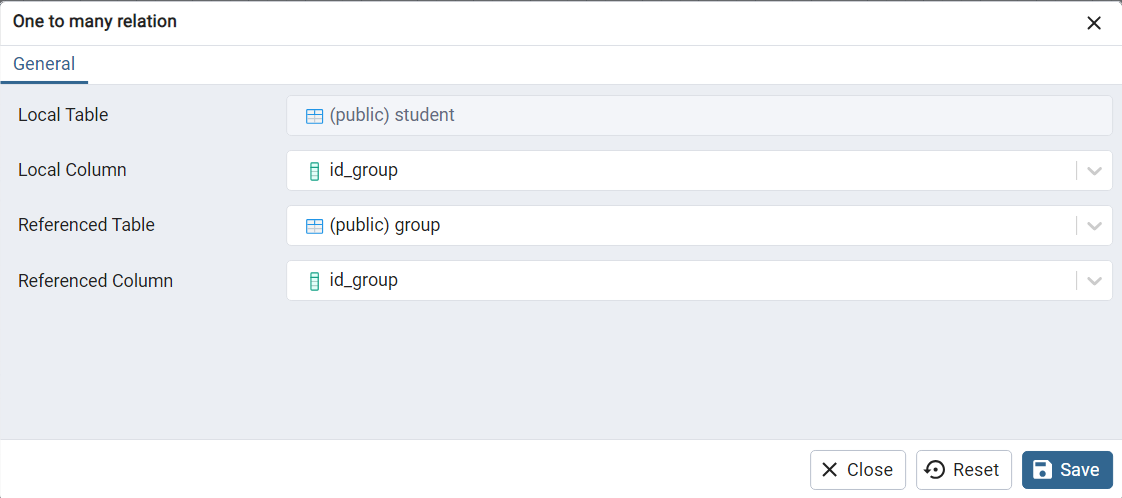
Перед тем, как вводить поле, нажмите на «+» в верхнем правом углу вкладки. Отметим, что id\_group будет первичным ключом, а название группы не может быть NULL (название группы должно быть введено, но это ограничение не предотвращает ввода пустой строки) и сохраним.



Аналогичным образом введем остальные таблицы, кроме group\_schedule.

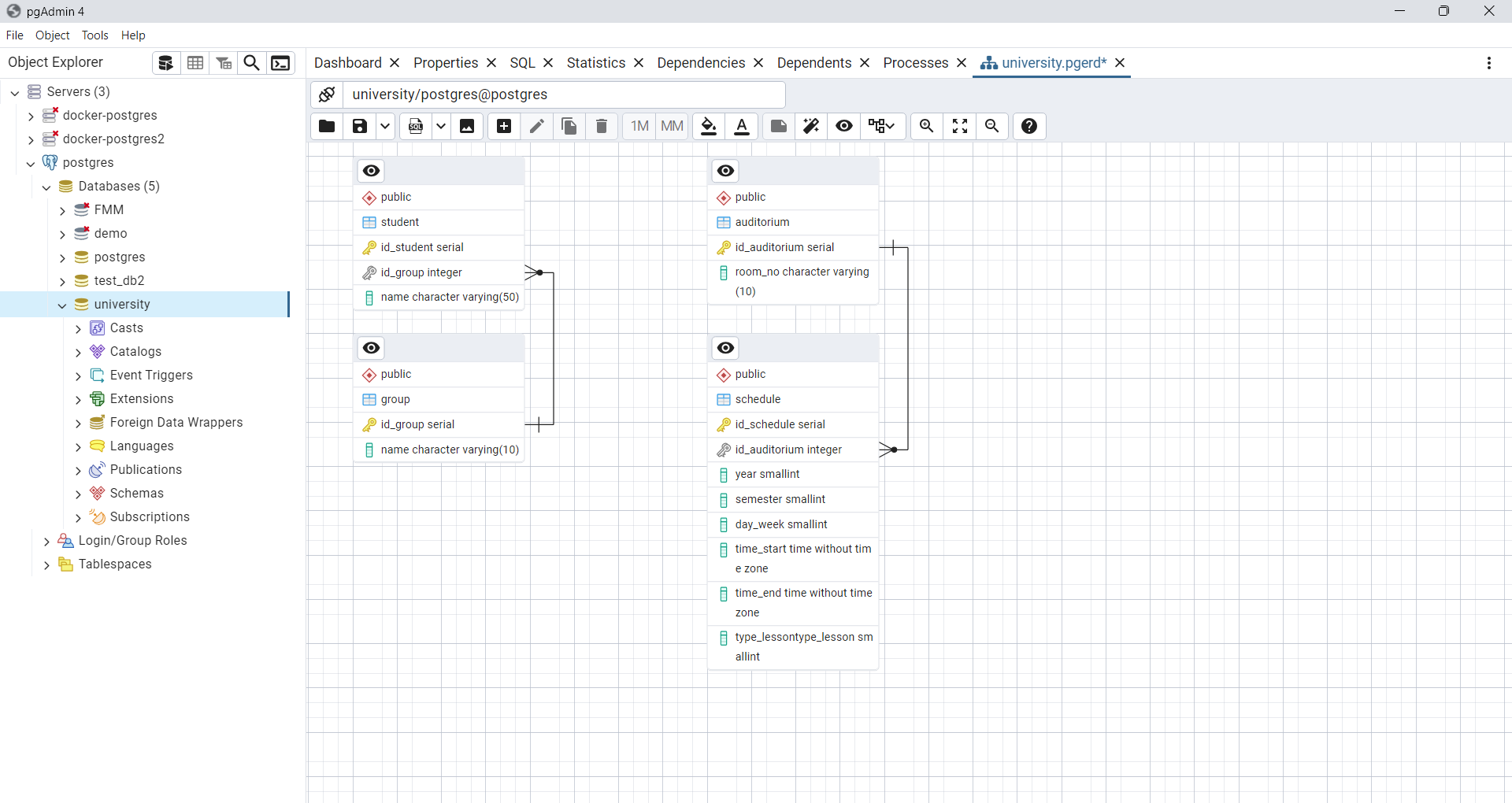


Организуем связь, между таблицами student и group. Щелкнем по таблице student, а затем нажмем на панели инструментов значок 1M (один-ко-многим). Откроется диалог связей один-ко-многим:

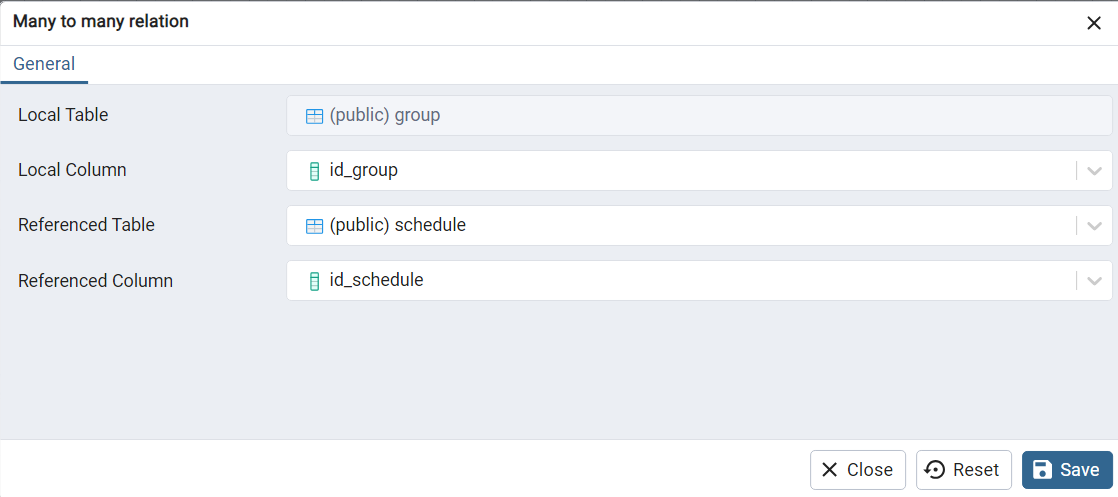


Заполним поля, выбирая соответствующие значения из выпадающих списков и сохраним.

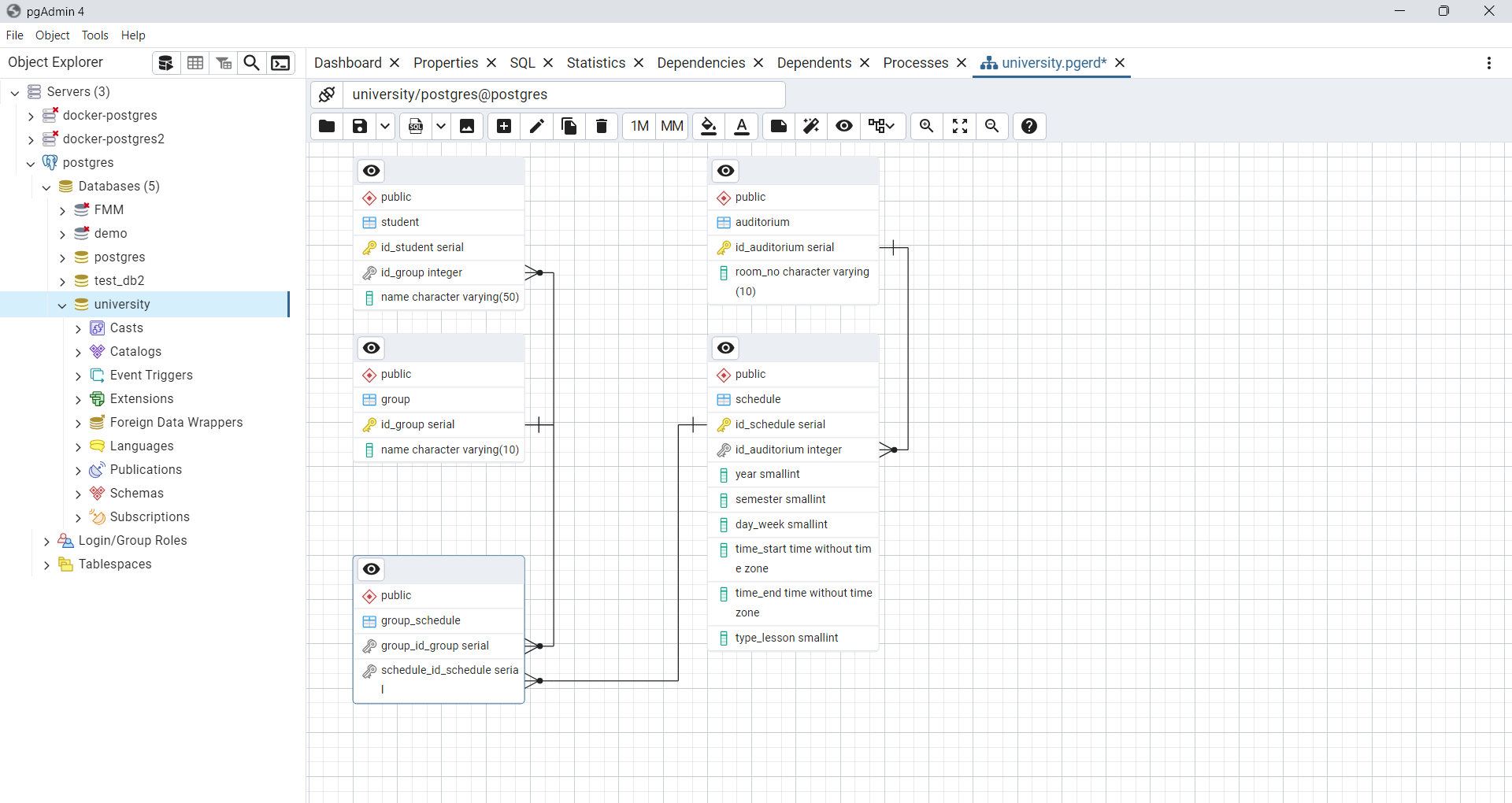
Так же организуем связь между schedule и auditorium:



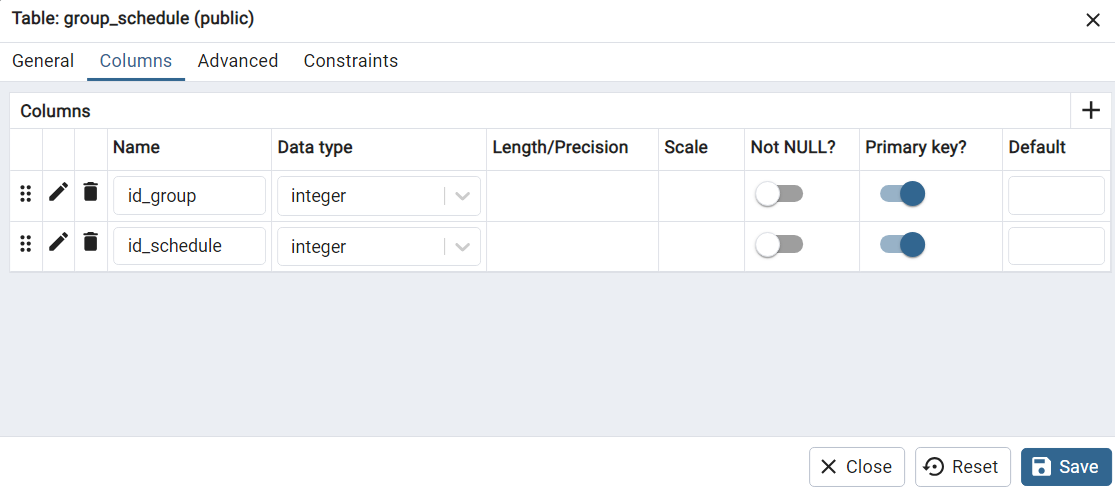
Можно было ввести таблицу group\_schedule и организовать соответствующие связи, но воспользуемся возможностью pgAdmin. Щелкнем по таблице group, а потом нажмем на значок связи MM (многие-ко-многим) на панели инструментов. Откроется соответствующий диалог:



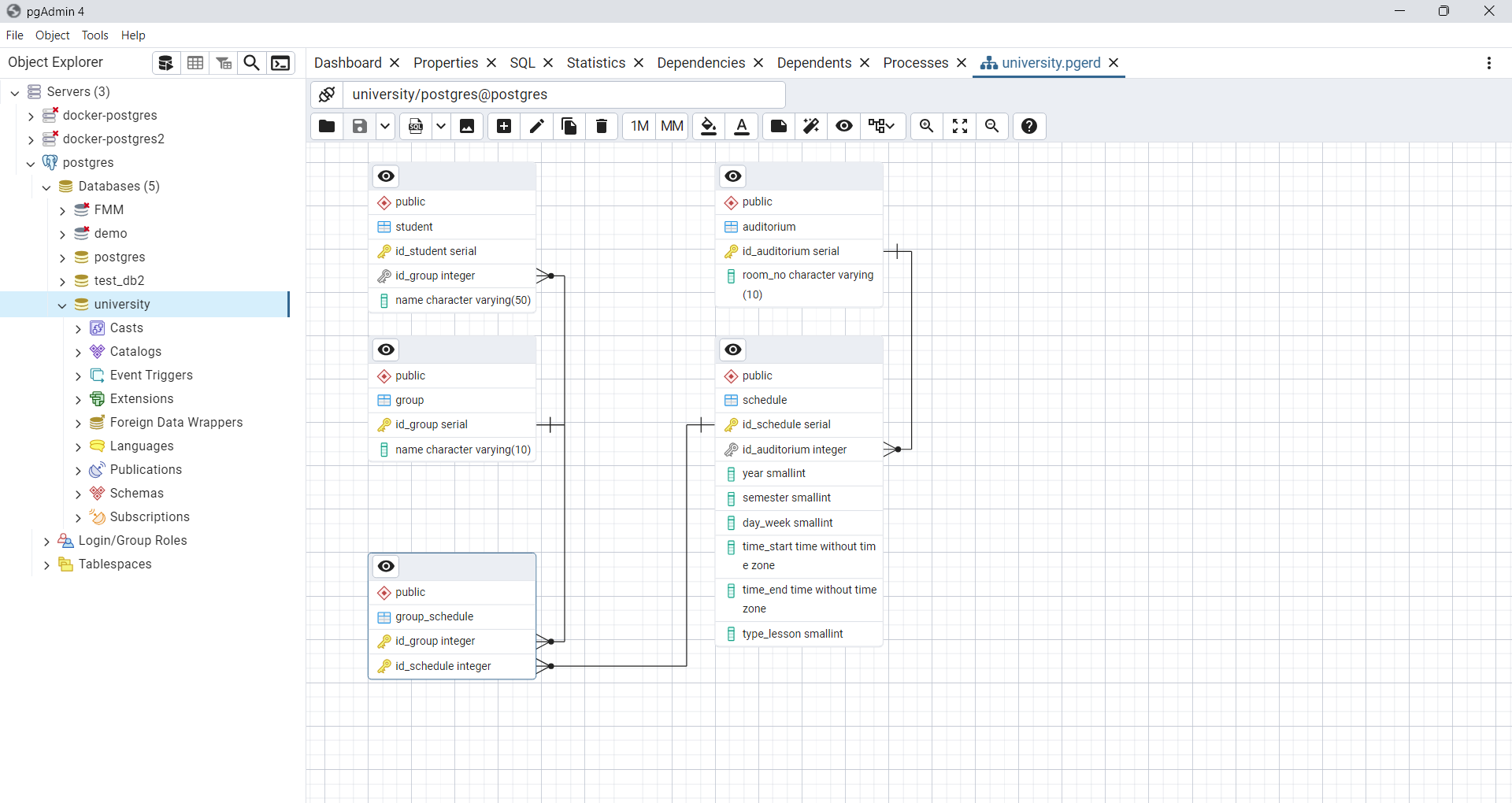
Заполним поля и сохраним.



Как видим, автоматически создалась ассоциативная таблица и связи с таблицами group и schedule. Название таблицы нас устраивает (алгоритм формирования имени таблицы очевиден). Название полей можно оставить, но удобно, когда имя поля внешний ключа совпадал с именем поля первичного ключа. А вот тип полей (serial) нас не устраивает. Отредактируем таблицу group\_schedule. Для этого дважды щелкнем по ней, отредактируем поля таблицы и создадим первичный ключ:



Сохраним и получаем:



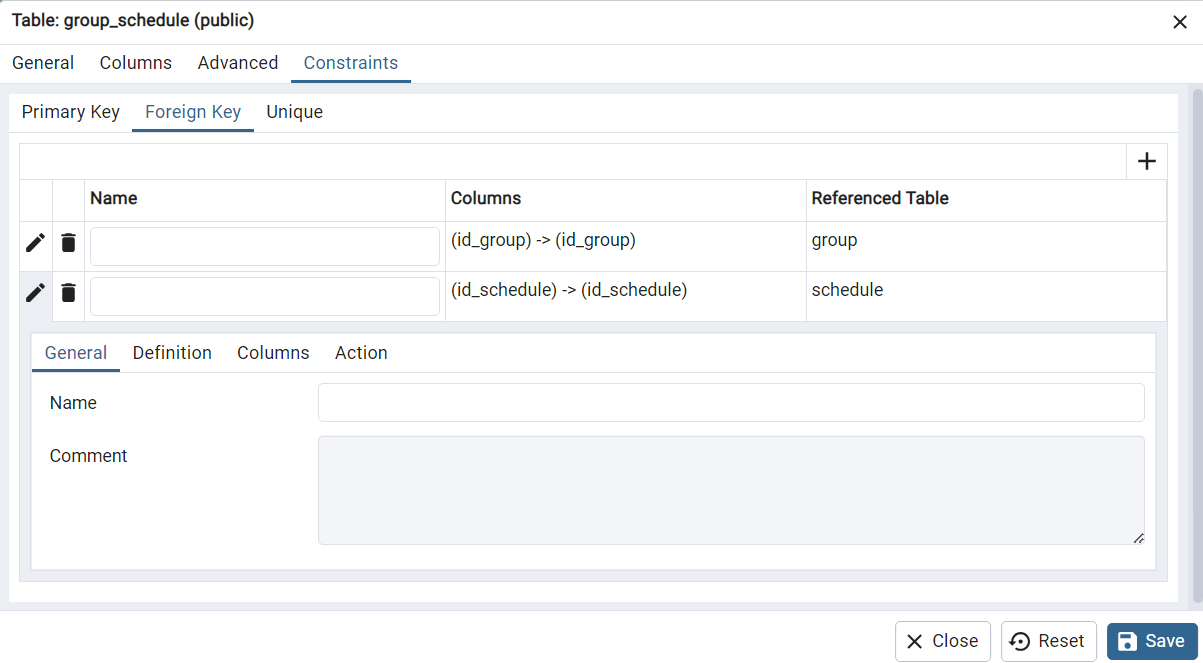
Сохраним ER–модель, нажав на значок дискета.

Все связи сформированы с опциями по умолчанию. Большинство опций можно оставить без изменения, только одну может потребоваться изменить. Рассмотрим следующею ситуацию. Клиент удаляет строку в таблице group, на эту строку ссылаются записи из таблицы student. В СУБД имеется возможность указать, что при удалении родительской записи (group) удалять дочерние записи (student), проставив опцию *CASCADE*, либо формировать ошибку указав *NO ACTION* (поведение по умолчанию). В этом случае таблицы group и student редко изменяются, поэтому, скорей всего, это упущение при разработке программы и допускать каскадное удаление будет ошибкой.

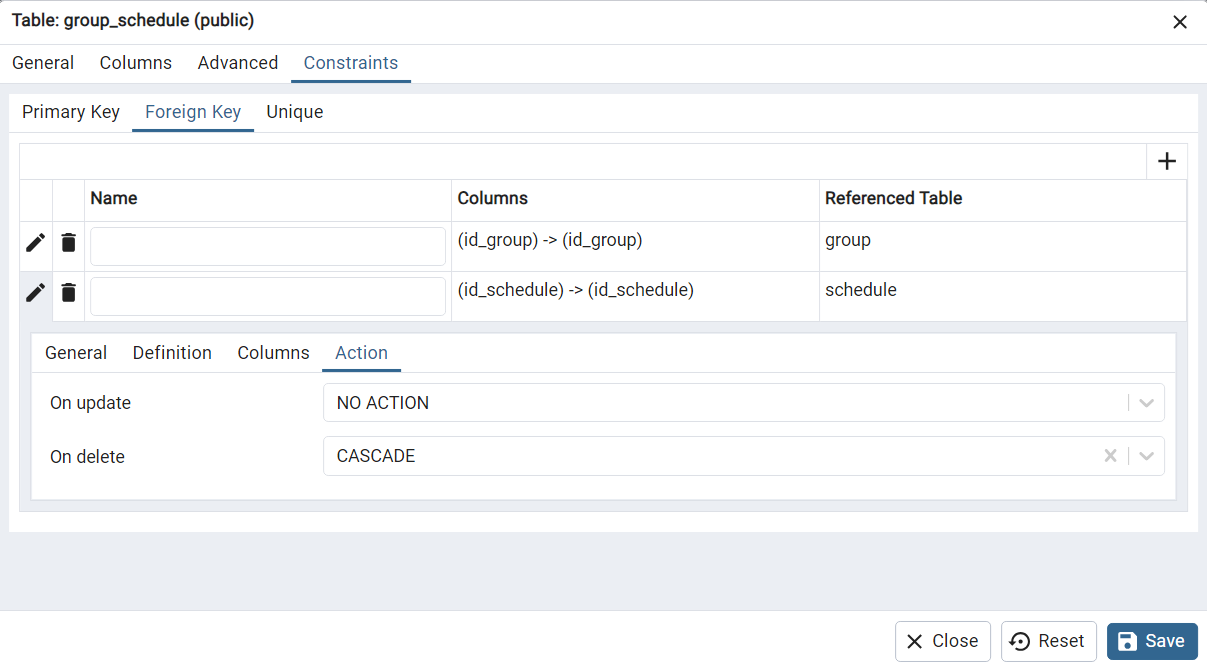
Рассмотрим таблицу schedule и ассоциативную таблицу group\_schedule. При формировании расписания перед началом семестра (даже при использовании программы) не все факторы могут быть учтены (например, лопнула труба и затопило аудиторию), приходится активно редактировать, удалять строки расписания. И если связь между таблицами schedule и group\_schedule имеет опцию *NO ACTION*, то перед тем, как удалить строку из group, надо удалить строки из group\_schedule, ссылающиеся на нее. Чтобы избежать дополнительной работы, достаточно выбрать опцию *CASCADE*.

*CASCADE* и *NO ACTION* наиболее часто используемые опции.

Отредактируем связь между таблицами schedule и group\_schedule. Дважды щелкнем по таблице group\_schedule. И в отрывшемся диалоге перейдем во вкладку *Constrains*, в ней выбираем вкладку *Foreign Key*, в ней нажимаем на значок редактирования связи schedule и group\_schedule:

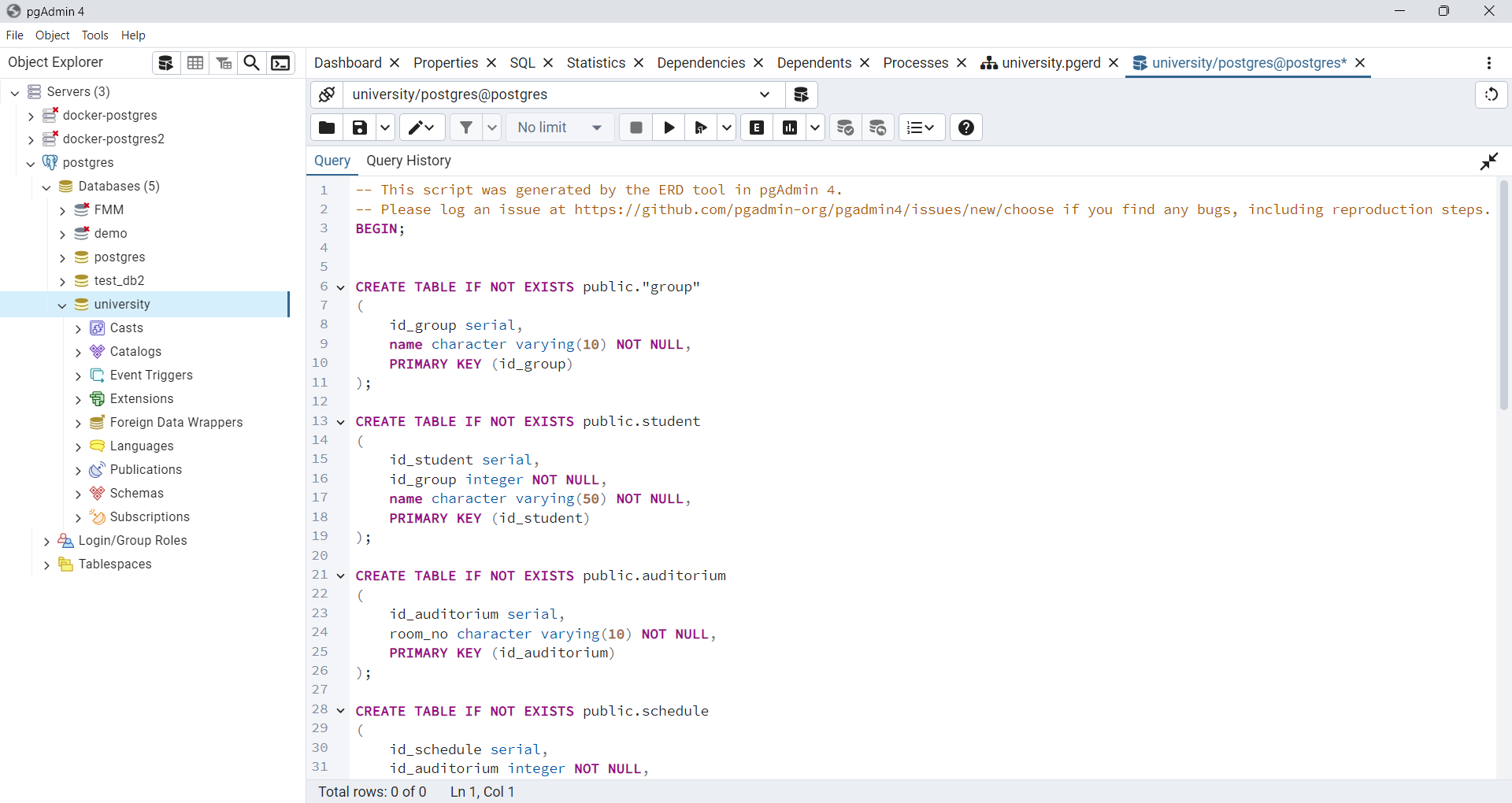


Далее, *Action* и для операции удаления выбираем – CASCADE:

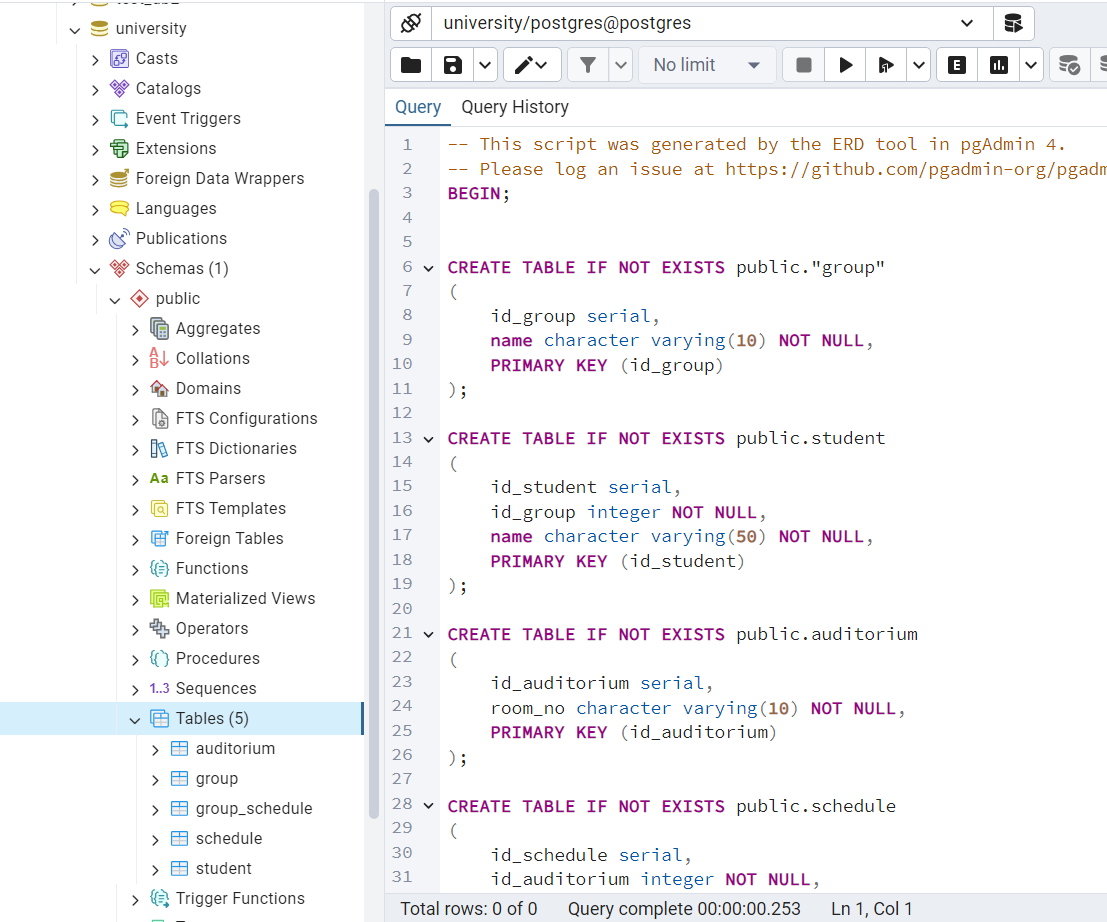


Сохраним изменения.

Построив ER–модель, обсудив с коллегами и учтя их замечания, приступаем к созданию таблиц в PostgreSQL. Возникает вопрос, неужели по второму разу вводить таблицы и связи? Конечно нет. На панели инструментов имеется значок  (Generate SQL), нажав который сформируется SQL-скрипт, содержащий наши труды по разработке БД:



Запустим SQL-скрипт. И БД university сформируются таблицы, соответствующие нашей ER–модели.



## 3. Практическое задание

В данной лабораторной работе вам предстоит разработать ER–модель базы данных системы, предметная область которой кратко описана в Приложении. Конкретную предметную область можно получить, взяв вариант, совпадающий с вашим порядковым номером в списке группы.

Для разработки ER–модели можно использовать любую предназначенную для этой цели программу.

Необходимо определить первичные и внешние ключи (с атрибутами NO ACTION, CASCADE).

## 4. Список использованных источников

1. Postgres Pro Standard Документация.

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

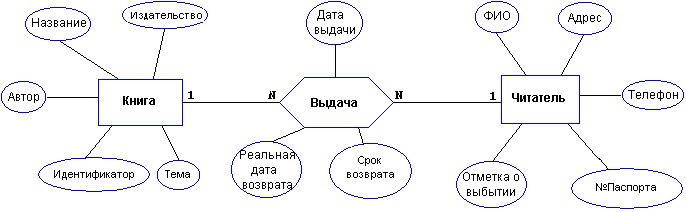
1. pgAdmin Documentation.

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

## 5. Приложение. Схемы предметных областей

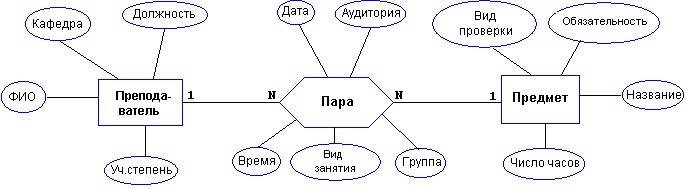
Вариант №1

Предметная область для практических заданий: **Библиотека**



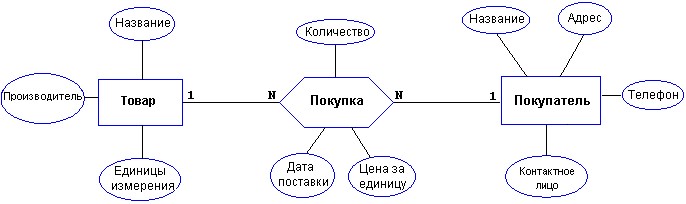
Вариант №2

Предметная область для практических заданий: **Университет**



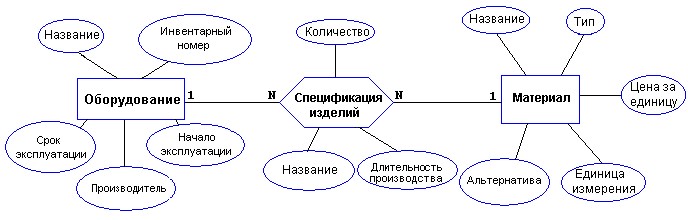
Вариант №3

Предметная область для практических заданий: **Отдел продаж**



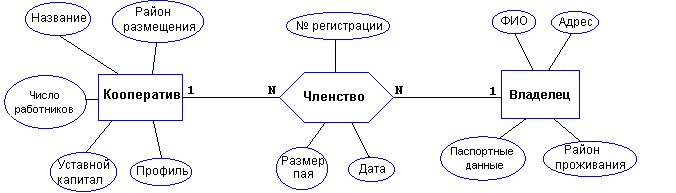
Вариант №4

Предметная область для практических заданий: **Производство**



Вариант №5

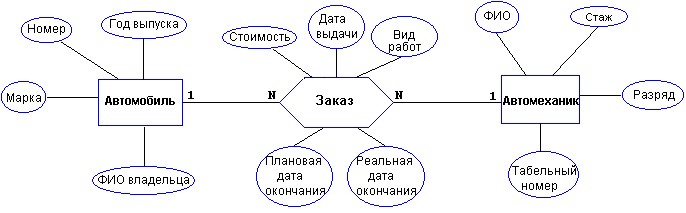
Предметная область для практических заданий: **Кооперативы**



Примечание: профиль - продуктовый, галантерейный, канцелярский и т.п.

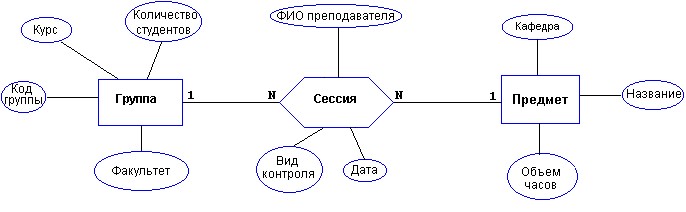
Вариант №6

Предметная область для практических заданий: **Автомастерская**



Вариант №7

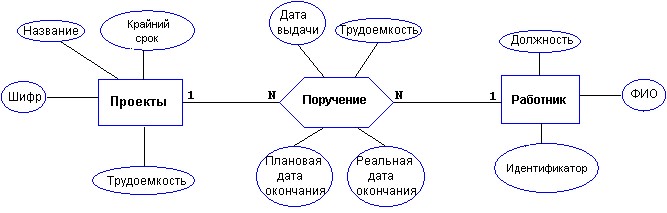
Предметная область для практических заданий: **Сессия**



Вид контроля - зачет, экзамен.

Вариант №8

Предметная область для практических заданий: **Управление проектом**

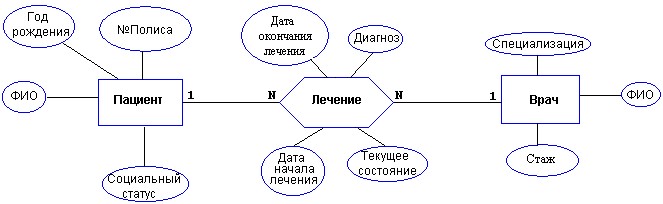


Категория дисциплины - гуманитарная, математическая, компьютерная и т.д.

Вид контроля - зачет, экзамен.

Вариант №9

Предметная область для практических заданий: **Поликлиника**

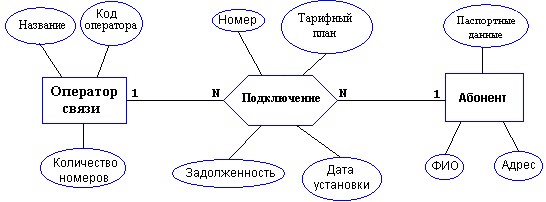


Текущее состояние - средней тяжести, тяжелое, направлен в стационар, умер. Социальный статус пациента - учащийся, работающий, врем. неработающий, инвалид, пенсионер

Специализация врача - терапевт, хирург и т.п.

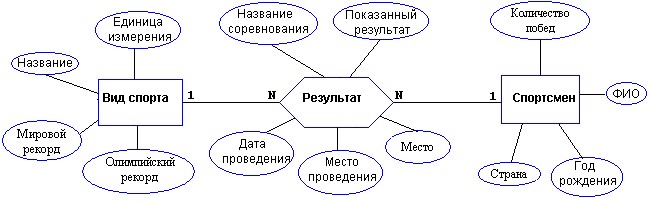
Вариант №10

Предметная область для практических заданий: **Сотовая связь**



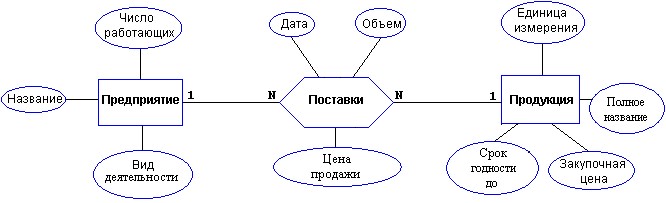
Вариант № 11

Предметная область для практических заданий: **Спорт**



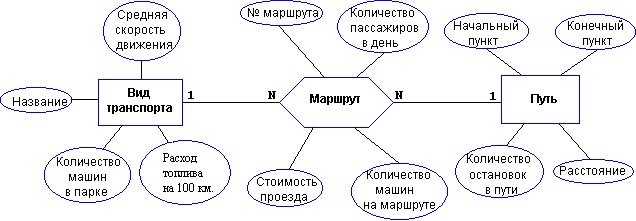
Вариант №12

Предметная область для практических заданий: **Поставки**



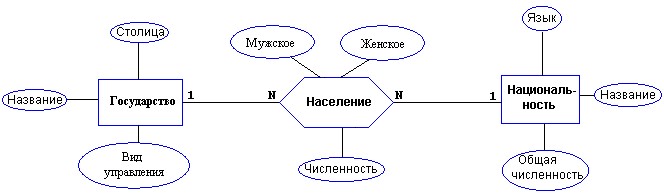
Вариант №13

Предметная область для практических заданий: **Транспорт**



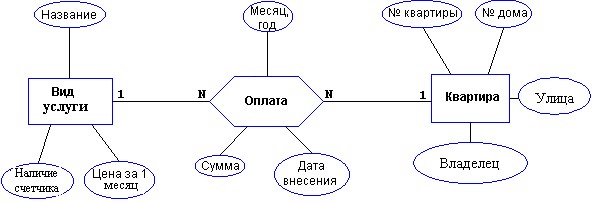
Вариант № 14

Предметная область для практических заданий: **География**



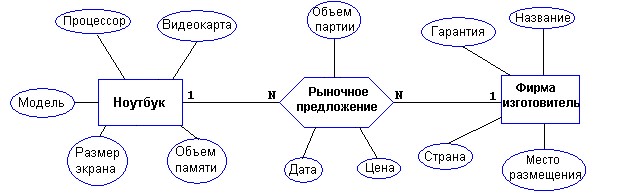
Вариант №15

Предметная область для практических заданий: **Домоуправление**



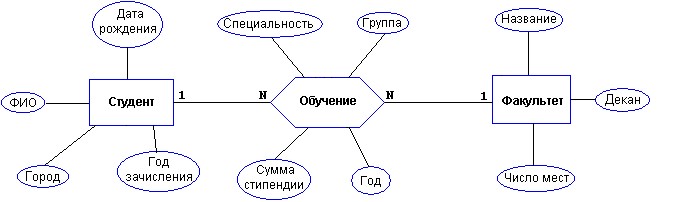
Вариант №16

Предметная область для практических заданий: **Ноутбуки**



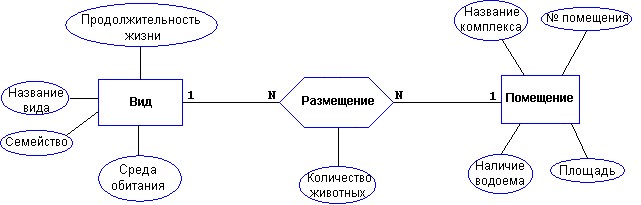
Вариант №17

Предметная область для практических заданий: **Деканат**



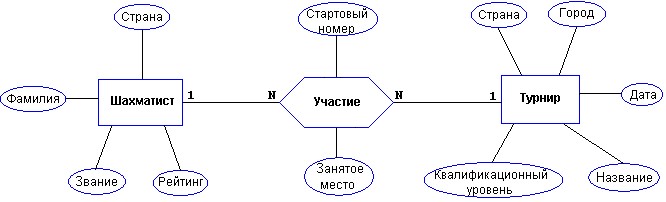
Вариант №18

Предметная область для практических заданий: **Зоопарк**



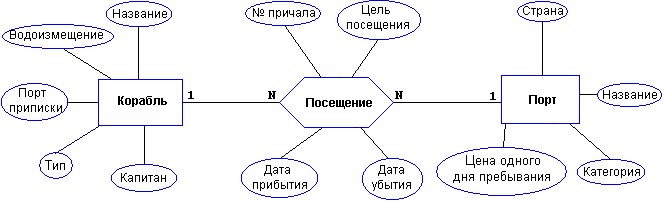
Вариант №19

Предметная область для практических заданий: **Шахматы**



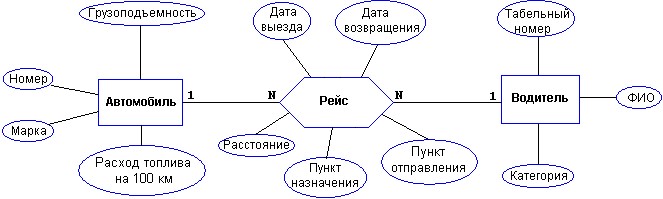
Вариант №20

Предметная область для практических заданий: **Судоходство** 1



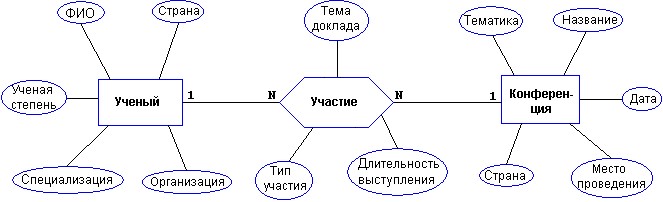
Вариант №21

Предметная область для практических заданий: **Грузоперевозки**



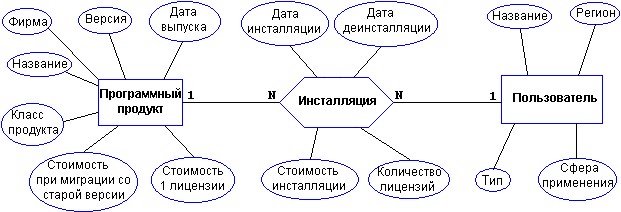
Вариант №22

Предметная область для практических заданий: **Научные конференции**



Вариант №23

Предметная область для практических заданий: **Программные продукты**

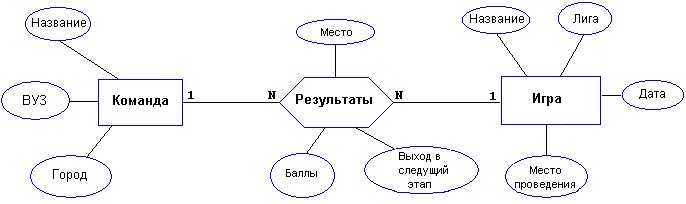


Класс: ОС, сервер приложений, СУБД, Web-сервер, система программирования и т.д.

Тип пользователя: индивидуальный, корпоративный, совместный, групповой и др.

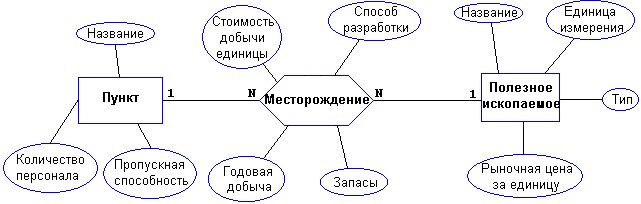
Вариант №24

Предметная область для практических заданий: **КВН**



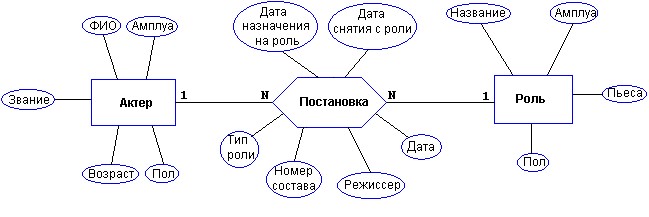
Вариант №25

Предметная область для практических заданий: **Добыча ресурсов**



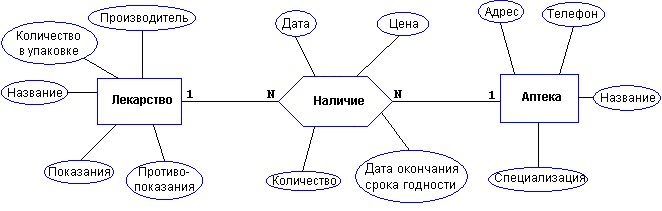
Вариант №26

Предметная область для практических заданий: **Театр**



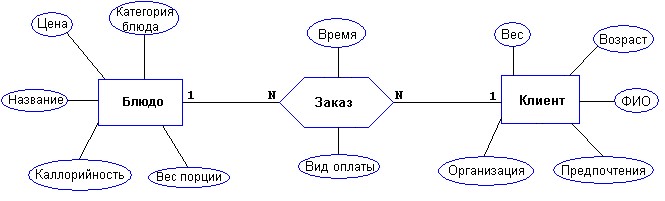
Вариант №27

Предметная область для практических заданий: **Справочная аптек**



Вариант №28

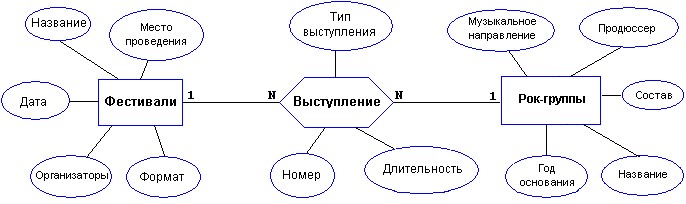
Предметная область для практических заданий: **Столовая**



Категория блюда: первое, второе, гарнир, десерт и т.д.

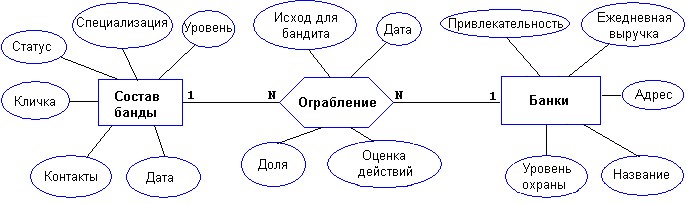
Вариант №29

Предметная область для практических заданий: **Рок-группы**



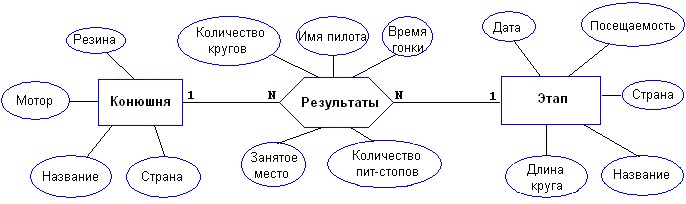
Вариант №30

Предметная область для практических заданий: **ОПГ**



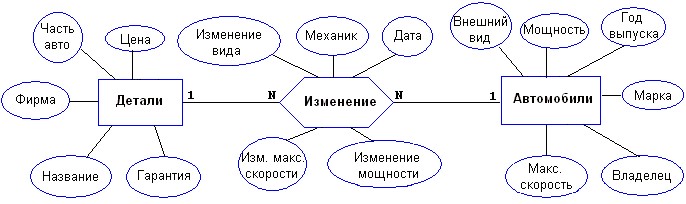
Вариант №31

Предметная область для практических заданий: **Формула 1**



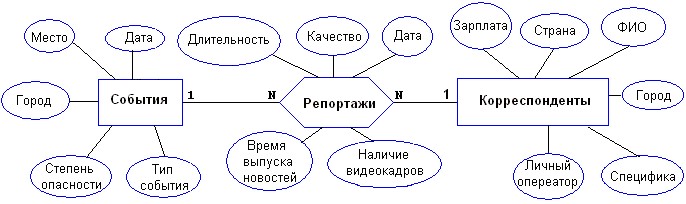
Вариант №32

Предметная область для практических заданий: **Тюнинг**



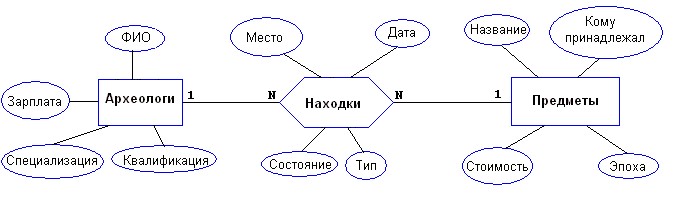
Вариант №33

Предметная область для практических заданий: **Тележурналистика**



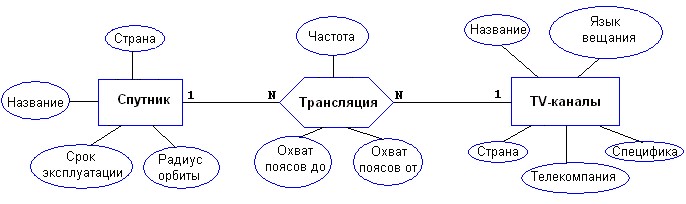
Вариант №34

Предметная область для практических заданий: **Археология**



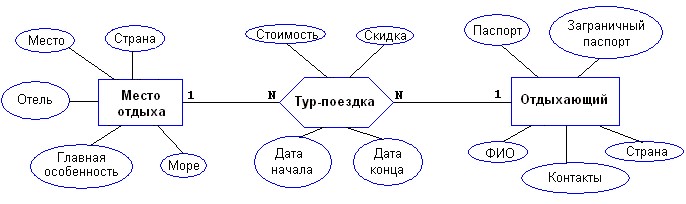
Вариант №35

Предметная область для практических заданий: **Телевещание**



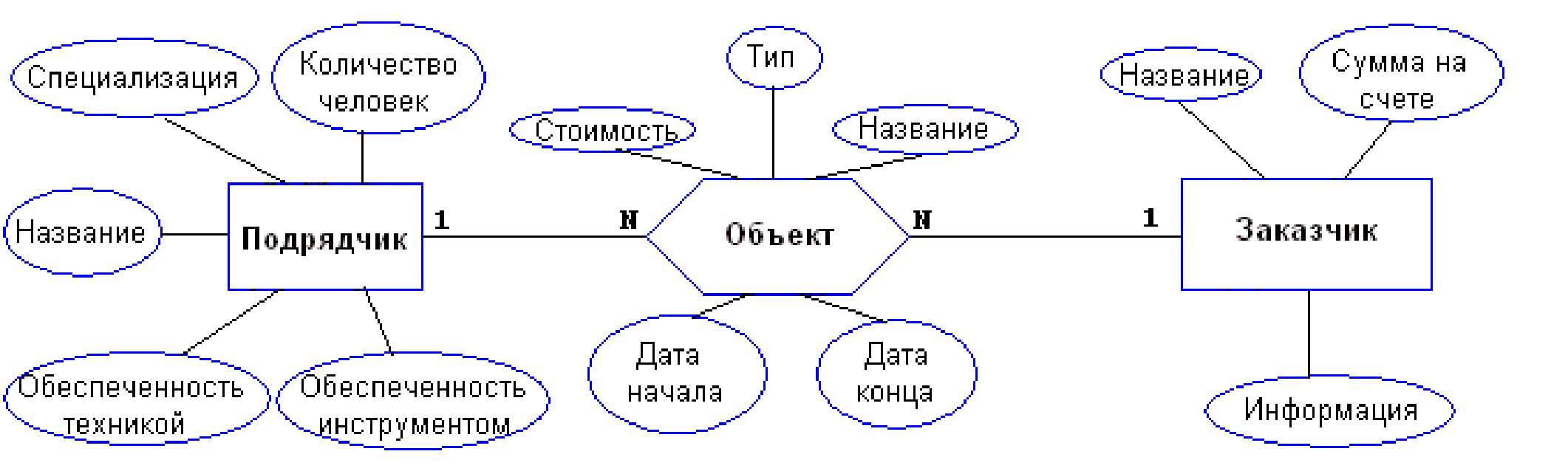
Вариант №36

Предметная область для практических заданий: **Турфирма**

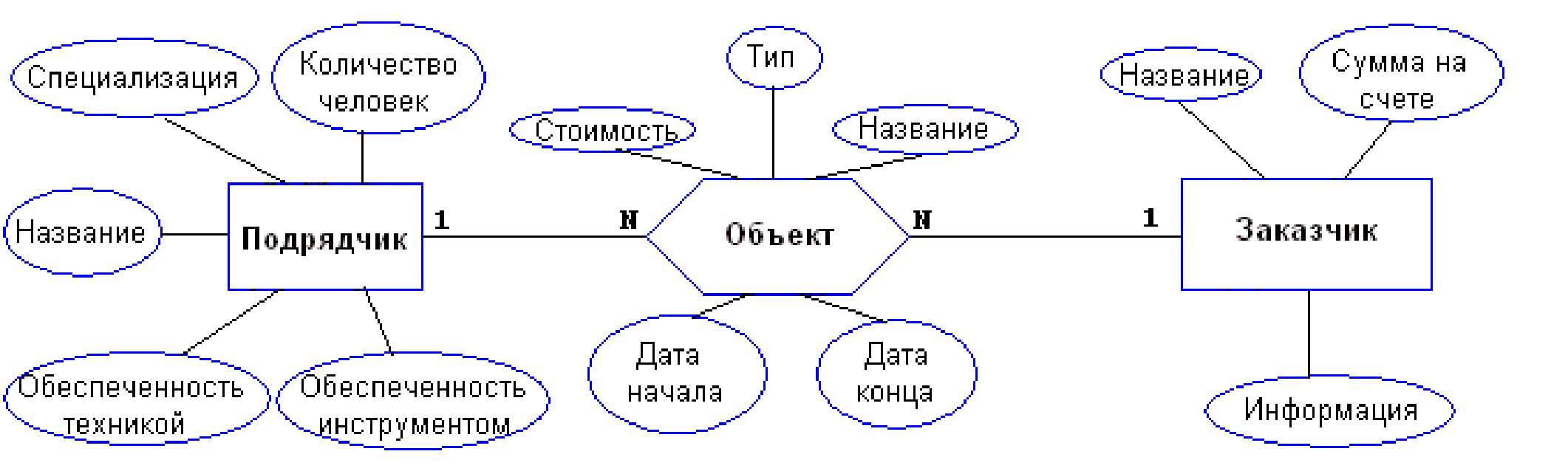


Вариант №37

Предметная область для практических заданий: **Строительство**



Тип объекта: промышленный, частный, специальный, хозяйственный и.т.д

Вариант №38

Предметная область для практических заданий: **Искусство**

Тип произведения: скульптура, живопись, литье, графика и т.д.

Тип места хранения: частная коллекция, музей, галерея, неизвестно (тогда все остальные параметры пустые) и т.д.

1. В современных БД при соответствующих настройках связей между таблицами изменения первичного клича приводит к изменению внешнего ключа дочерней таблицы. Однако, пренебрегать временем, затраченным на выполнение этой операции тоже не следует. [↑](#footnote-ref-1)
2. Все занятия будут проходить в одном здании. [↑](#footnote-ref-2)