

Лабораторная работа №7

Проектирование и создание объектов базы данных

Техническая документация по управлению таблицами на языке SQL (определение данных): <https://postgrespro.ru/docs/postgrespro/15/ddl>

Типы данных: <https://postgrespro.ru/docs/postgrespro/15/datatype#DATATYPE-TABLE>

1. Проектирование и создание базы данных

Проектирование базы данных в общем случае состоит из двух этапов: логического и физического. Во время логического проектирования разрабатывается модель базы данных, не зависящая от конкретной СУБД, но в терминологии конкретной даталогической модели (например, реляционной). В процессе же физического проектирования создается модель базы данных, оптимизированная для реальной СУБД.

1.1. Логическая модель БД «Факультет» (BD «Faculty»)

На одном из факультетов некоторого учебного заведения (например, ВУЗа) проводятся учебные занятия.

Необходимо создать базу данных (БД), в которой будет вестись учет сведений об учебных занятиях: какие занятия ведутся, у каких учебных групп (в т. ч. учитывается состав группы), кто из преподавателей проводит занятия.

Выделим следующие **объекты (таблицы)** с их характеристиками (атрибутами), также укажем требования к атрибутам.

Группа (SGroup)

- *Код группы* – обязательный атрибут, содержащий уникальный идентификатор учебной группы согласно классификации, принятой в ВУЗе, первичный ключ (например, МКб-4301..., ФИБ-2301...);
- *Направление подготовки* – необязательный атрибут;
- *Профиль* – необязательный атрибут;
- *Курс* – необязательный атрибут, целое число от 1 до 5.

Студент (Student)

- *Номер зачетной книжки* – обязательный атрибут, целое **шестизначное число**, первичный ключ;
- *Фамилия* – обязательный атрибут;
- *Имя* – обязательный атрибут;
- *Отчество* – необязательный атрибут;
- *Пол* – необязательный атрибут, предполагает ввод одного из двух возможных значений «ж» или «м»;
- *Дата рождения* – необязательный атрибут, **не может быть больше сегодняшней даты¹**;
- *Адрес* – необязательный атрибут, если не указан, то по умолчанию **устанавливается значение «неизвестно»**;
- *Сотовый телефон* – необязательный атрибут, предполагает **ввод символов в формате: +70000000000**, где 0 – любая цифра.
- *E-mail* – необязательный атрибут;

¹ Текущую дату можно получить с помощью одной из функций current_date или Now()

- *Год поступления* – необязательный атрибут, целое **четырёхзначное** число;
- *Проживает в общежитии* – обязательный атрибут, предполагает ввод одного из значений «да» или «нет», по умолчанию устанавливается значение «нет»;

Преподаватель (Teacher)

- *Код преподавателя* – обязательный атрибут, **генерируемый автоматически**, первичный ключ;
- *Фамилия* – обязательный атрибут;
- *Имя* – обязательный атрибут;
- *Отчество* – необязательный атрибут;
- *Сотовый телефон* – необязательный атрибут, предполагает **ввод символов в формате: +70000000000**, где 0 – любая цифра.
- *E-mail* – необязательный атрибут;
- *Кафедра* – обязательный атрибут;
- *Должность* – обязательный атрибут (например, ассистент, преподаватель, старший преподаватель, доцент, профессор);
- *Ученая степень* – необязательный атрибут (например, кандидат наук, доктор наук);
- *Дата принятия на работу* – обязательный атрибут, **не превосходит текущую дату, по умолчанию устанавливается текущая дата**;
- *Дата рождения* – необязательный атрибут, **не может быть больше сегодняшней даты**.

Дисциплина (Subject)

- *Код дисциплины* – обязательный атрибут, **генерируемый автоматически**, первичный ключ;
- *Название* – обязательный атрибут;
- *Количество лекционных часов* – обязательный атрибут, целое **неотрицательное** число;
- *Количество практических часов* – обязательный атрибут, целое **неотрицательное** число;
- *Количество лабораторных часов* – обязательный атрибут, целое **неотрицательное** число;
- *Общий объем часов* – обязательный генерируемый атрибут, **вычисляется по формуле** как сумма количества лекционных, практических и лабораторных часов;
- *Семестр изучения* – обязательный атрибут, целое число **от 1 до 10**;
- *Форма контроля* – обязательный атрибут, предполагает ввод одного из возможных значений «зачет», «экзамен» или «зачет + экзамен».

Аудитория (Classroom)

- *№ Кабинета* – обязательный атрибут, первичный ключ (например, 16-401);
- *Количество мест* – обязательный атрибут, целое **положительное** число;
- *Компьютерный класс* – обязательный атрибут, предполагает ввод одного из значений «да» или «нет», по умолчанию устанавливается значение «нет»;
- *Проектор* – обязательный атрибут, предполагает ввод одного из значений «да» или «нет», по умолчанию устанавливается значение «нет».

Занятие (Lesson)

- *Группа* – обязательный атрибут, входит в состав первичного ключа;

- *Дисциплина* – обязательные атрибуты, входит в состав первичного ключа;
- *Вид занятия* – обязательный атрибут, входит в состав первичного ключа;
- *Преподаватель* – обязательный атрибут;
- *Аудитория* – обязательный атрибут.

Успеваемость (Mark)

- *№Зачетки* – обязательный атрибут, входит в состав первичного ключа;
- *Дата* – обязательный атрибут, входит в состав первичного ключа;
- *Дисциплина* – обязательный атрибут, входит в состав первичного ключа;
- *Преподаватель* – обязательный атрибут.
- *Оценка* – обязательный атрибут (например, «зачтено», «не зачтено», «5», «4», «3», «2», «неявка»).

При этом нужно учитывать следующие **ограничения (связи)**:

- у каждой группы может проводиться несколько занятий;
- в каждой группе учится несколько студентов;
- каждый преподаватель может проводить несколько занятий;
- одна и та же дисциплина может проводиться разными преподавателями у разных групп и в различных формах (лекция, семинар, лабораторное занятие и т. п.);
- у каждого студента может быть несколько оценок по одной или разным дисциплинам;
- занятие на постоянной основе проводится в одном кабинете.

1.2. Физическая модель

При переходе от логической модели к физической необходимо обеспечить **целостность данных**, под которой понимается согласованность и точность информации в базе данных. Поэтому, прежде чем рассмотреть непосредственно создание таблиц БД, рассмотрим средства обеспечения целостности данных.

1.2.1. Обеспечение целостности данных

Существуют общие и специфические **правила целостности** (первые два – общие):

Целостность объектов: требует наличия в каждой таблице первичного ключа, значения которого не должны быть пустыми.

Ссылочная целостность: гарантирует поддержание постоянной связи между первичным ключом и внешним ключом.

Целостность области значений (доменная целостность): определяет набор допустимых для поля значений, в том числе и допустимость значений NULL.

Целостность данных может быть обеспечена двумя способами:

1. **Декларативным.** Критерии, которым должны удовлетворять данные, задаются при определении объекта и являются частью определения базы данных.

Преимущества: Контроль целостности выполняется автоматически СУБД.

2. **Процедурным.** Критерии описываются в пакетах операторов, выполнение которых и определяет целостность данных.

Преимущества: реализуется как на клиенте, так и на сервере с помощью различных программных средств.

Декларативная целостность реализуется при помощи **ограничений**, которые являются наиболее рекомендуемыми для обеспечения целостности данных. Различные типы целостности обеспечиваются соответствующими ограничениями:

- **PRIMARY KEY** – определение поля или группы полей в качестве первичного ключа, т.е. позволяет уникально идентифицировать каждую запись таблицы, в этом случае недопустимы повторяющиеся и неопределенные (NULL) значения.
- **FOREIGN KEY** – определение поля или группы полей в качестве внешнего ключа, т.е. позволяет устанавливать ссылку на первичный ключ другой таблицы, содержащий идентичные данные. Данные в этом поле могут принимать значения, определенные в соответствующем первичном ключе, либо значения NULL.
- **UNIQUE** – определяет уникальность данных в некотором поле таблицы, причем данное ограничение допускает значения NULL. В одной таблице разрешено применение несколько таких ограничений, поля с таким ограничением могут быть использованы в качестве внешних ключей.
- **DEFAULT** – указывает значение поля по умолчанию, если значение явно не указано при вставке данных. Для каждого поля можно применить только одно такое ограничение.
- **CHECK** – ограничение-проверка на значение поля в виде логического выражения (проверка истинности), что позволяет определить диапазон допустимых значений поля. Причем данное ограничение позволяет ссылаться на значения других полей.

1.2.2. Создание таблицы

Таблица базы данных может быть создана либо средствами Transact-SQL, либо с помощью специальных интерфейсных средств. Так же, как и для создания базы данных, рассмотрим сначала синтаксис создания таблиц на SQL.

Создание новой таблицы в базе данных командой **CREATE TABLE**:

```
CREATE TABLE имя_таблицы
    ( [ имя_поля тип_данных | домен
      [ ограничение_столбца [ ... ] ]
      | ограничение_таблицы
      [ , ... ] ] )
```

ограничение_столбца определяется выражением:

```
CONSTRAINT имя_ограничения ]
{ NOT NULL | NULL
| CHECK ( выражение )
| DEFAULT выражение_по_умолчанию },
```

ограничение_таблицы – выражением:

```
CONSTRAINT имя_ограничения ]
{ CHECK ( выражение )
| UNIQUE ( имя_поля [ , ... ] )
| PRIMARY KEY ( имя_поля [ , ... ] )
| FOREIGN KEY ( имя_поля [ , ... ] )
  REFERENCES родит_таблица [(ключ_поле [ , ... ])]
[ ON DELETE действие ]
[ ON UPDATE действие ] }
```

Определения имени и столбцов таблицы описываются следующим образом:

имя_таблицы – имя таблицы, как правило, дополненное именем схемы.

имя_поля – уникально в пределах таблицы и удовлетворяет правилам именования объектов БД.

тип_данных – показывает, данные какого вида хранятся в столбце.

домен – имя домена, т. е. ранее описанного типа столбца.

Ограничения поля (столбца):

CONSTRAINT – с этого ключевого слова должно начинаться наложение ограничения на поле. После **CONSTRAINT** указывается имя ограничения, если оно не указано, то система генерирует имя автоматически и является при этом неосмысленным. Поэтому лучше самим давать ограничениям осмысленные имена. После имени ограничения следуют определения ограничения, описанные ниже:

NULL / NOT NULL – разрешает или запрещает использование в поле пустые значения. Данный вид ограничения, как правило, используется без конструкции **CONSTRAINT имя_ограничения**. Например,
name text NOT NULL;

CHECK (выражение) – задается выражение, возвращающее булевский результат, по которому определяется, будет ли успешна операция добавления или изменения для конкретных строк.

ВАЖНО: ограничение-проверка, заданное как ограничение столбца, должно ссылаться только на значение самого столбца, тогда как ограничение на уровне таблицы может ссылаться и на несколько столбцов.

DEFAULT выражение_по_умолчанию – задает значение по умолчанию для поля, в определении которого оно присутствует. Это выражение будет использоваться во всех операциях добавления данных, в которых не задается значение данного поля. Например,
modtime timestamp DEFAULT current_timestamp;

Для поля можно определить больше одного ограничения. Для этого их нужно просто указать одно за другим:

price numeric NOT NULL CHECK (price > 0);

Ограничения таблицы:

В SQL предусмотрены также возможности определения ограничений для таблиц в целом. К таким ограничениям относятся первичные и внешние (вторичные) ключи таблицы.

PRIMARY KEY – задает поля, которые входят в состав первичного ключа. СУБД автоматически следит за тем, чтобы РК был уникален. Кроме того, в определениях полей РК должно быть указано, что они не могут содержать значения NULL.

FOREIGN KEY – задает внешний ключ таблицы, с помощью которой определяется связь родительской и дочерних таблиц БД.

имя_поля [, ...] – определяет поля (столбцы) дочерней (создаваемой) таблицы, которые образуют FK;

родит_таблица [(ключ_поле [, ...])] – определяет таблицу, в которой описан РК, на который ссылается FK дочерней таблицы для обеспечения ссылочной целостности.

В **FOREIGN KEY** задаются также правила удаления и обновления. **ON DELETE** задает действие при удалении некоторой строки во внешней таблице. **ON UPDATE** – при изменении значения в целевых столбцах внешней таблицы. Возможны следующие варианты действий:

NO ACTION – определяет запрет удаления/изменения родительской записи при наличии подчиненных записей в дочерней таблице. Этот вариант действия подразумевается по умолчанию;

RESTRICT – выдает ошибку, если при удалении или изменении записи произойдет нарушение ограничения внешнего ключа. Этот вариант подобен NO ACTION, но эта проверка будет не откладываемой;

CASCADE – при удалении записи родительской таблицы происходит удаление подчиненных записей в дочерней таблице; при изменении поля связи в записи родительской таблицы происходит изменение на то же значение поля FK у всех подчиненных записей в дочерней таблице;

SET NULL, SET DEFAULT – установить значение ссылающегося столбца в NULL или в значение по умолчанию (оно должно быть задано).

IDENTITY – позволяет создать автоинкрементный первичный ключ (порядковый номер):

```
CREATE TABLE имя_таблицы ( ...  
имя_столбца integer PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY  
);
```

Создать целочисленный столбец со значением по умолчанию, извлекаемым из генератора последовательности также можно с помощью специальной короткой записи:

```
CREATE TABLE имя_таблицы ( ...  
имя_столбца SERIAL  
);
```

Типы данных smallserial, serial и bigserial не являются настоящими типами, а представляют собой просто удобное средство для создания столбцов с уникальными идентификаторами. Эти типы данных реализованы через последовательности, поэтому в числовом ряду значений столбца могут образовываться пропуски (или «дыры»), даже если никакие строки не удалялись. Значение, выделенное из последовательности, считается «задействованным», даже если строку с этим значением не удалось вставить в таблицу. Это может произойти, например, при откате транзакции, добавляющей данные.

Чтобы вставить в столбец serial следующее значение последовательности, ему нужно присвоить значение по умолчанию. Это можно сделать, либо исключив его из списка столбцов в операторе INSERT, либо с помощью ключевого слова DEFAULT.

Генерируемое (вычисляемое) поле является полем особого рода, которое всегда вычисляется из других. Для создания генерируемого столбца нужно использовать предложение GENERATED ALWAYS AS команды CREATE TABLE. Ключевое слово STORED, определяющее тип хранения генерируемого столбца, является обязательным.

```
CREATE TABLE имя_таблицы (  
...,  
имя_поля тип_данных GENERATED ALWAYS AS (формула) STORED  
);
```

Рассмотрим пример создания таблицы, содержащей сведения о **Сотрудниках**:

```
CREATE TABLE Org.Сотрудник (  
    КодСотрудника serial NOT NULL PRIMARY KEY,  
    Фамилия varchar(30) NOT NULL,  
    Имя varchar(30) NOT NULL,  
    Отчество varchar(30) NULL,  
    Пол char(1) NOT NULL,  
    CONSTRAINT check_Gender CHECK (Пол IN ('м','ж')),  
    ДатаРождения date NOT NULL,
```

```

        CONSTRAINT check_BD CHECK (EXTRACT (year from AGE
(current_date, ДатаРождения))>=18),
        ДомашнийАдрес varchar(50) NULL,
        Телефон char(12) NOT NULL,
        CONSTRAINT check_Phone CHECK (Телефон ~ '^+7\d{10}$'),
        ДатаНайма date NOT NULL DEFAULT current_date,
        CONSTRAINT check_DH CHECK (ДатаНайма<=NOW()),
        Отдел varchar(10) NOT NULL,
        CONSTRAINT fk_IDEmpl FOREIGN KEY (Отдел)
REFERENCES Org.Отдел(КодОтдела)
ON UPDATE CASCADE
ON DELETE NO ACTION
);

```

Здесь для ограничения внешнего ключа дано имя `fk_IDEmpl`, а для ограничений на значения – `check_BD`, `check_Phone`, `check_DH`. Первичный ключ представляет собой автозаполняемое поле. В столбцы `Пол`, `ДатаРождения`, `Телефон` и `ДатаНайма` можно вводить только значения, определенные в проверочных ограничениях. В поле `ДатаНайма` по умолчанию подставляется текущая дата. `'^+7\d{10}$'` – регулярное выражение, допускающее ввод '+7' и 10 цифр. `AGE (current_date, ДатаРождения)` – вычитает даты. `EXTRACT (year from ...)` – извлекает количество лет.

1.2.3. Изменение структуры таблицы

Если при создании таблицы была допущена ошибка или изменились требования к структуре БД, можно удалить её и создать заново. Но это будет неудобно, если таблица уже заполнена данными, или если на неё ссылаются другие объекты БД (например, по внешнему ключу). Поэтому SQL предоставляет набор команд для модификации (структуры) таблицы: добавление / удаление столбцов, добавление / удаление ограничений, изменение значения по умолчанию и типов столбцов, переименование столбцов и таблиц. Изменение структуры таблицы аналогично ее созданию:

```

ALTER TABLE имя_таблицы
[ ADD COLUMN имя_поля тип_данных [ ограничение ]
| DROP COLUMN имя_поля [ CASCADE ]
| ADD CONSTRAINT имя_ограничения ...
| DROP CONSTRAINT имя_ограничения
| ALTER COLUMN имя_поля
| TYPE новый_тип | SET DEFAULT зн-е_по_умолч | SET NULL | NOT NULL
| DROP NOT NULL
| DROP DEFAULT ]
[ | RENAME COLUMN старое_имя_поля TO новое_имя_поля ]
[ | RENAME TO новое_имя_таблицы ]
]

```

Параметры:

ADD COLUMN – добавления нового поля в таблицу, после него описываются параметры нового поля.

DROP COLUMN – удаление поля. Вместе со столбцом удаляются и включающие его ограничения таблицы. Однако если на столбец ссылается ограничение FK другой таблицы, СУБД не удалит это ограничение неявно. Разрешить удаление всех зависящих от этого столбца объектов можно, добавив указание **CASCADE**.

ADD CONSTRAINT – добавляется ограничение.

DROP CONSTRAINT – из таблицы удаляется ограничение с указанным именем.

DROP NOT NULL – удалить ограничение NOT NULL, т.к. у него нет имени, аналогично удалить значение по умолчанию – **DROP DEFAULT**.

ALTER COLUMN – изменение поля таблицы:

TYPE *новый_тип* – преобразование поля в другой тип данных. Будет успешно выполнено, только если все существующие значения в столбце могут быть неявно приведены к новому типу.

SET DEFAULT *значение_по_умолчанию* – назначить столбцу новое значение по умолчанию. Не влияет на существующие строки таблицы, задаёт значение по умолчанию для последующих команд INSERT.

SET NOT NULL – добавить ограничение NOT NULL, которое нельзя записать в виде ограничения таблицы. Ограничение проходит проверку автоматически и будет добавлено, только если ему удовлетворяют данные таблицы.

RENAME COLUMN *старое_имя_поля* **TO** *новое_имя_поля* – переименование столбца таблицы.

RENAME TO *новое_имя_таблицы* – переименование таблицы.

1.2.4. Удаление таблицы

Для удаления таблицы из БД средствами SQL необходимо выполнить команду:

DROP TABLE [IF EXISTS] *имя_таблицы* [, ...] [CASCADE | RESTRICT]

Параметры:

IF EXISTS – не считать ошибкой, если таблица не существует. В этом случае будет выдано замечание.

Имя_таблицы – имя таблицы (возможно, дополненное схемой), подлежащей удалению.

CASCADE – автоматически удалять объекты, зависящие от данной таблицы (например, представления), и, в свою очередь, все зависящие от них.

RESTRICT – отказать в удалении таблицы, если от неё зависят какие-либо объекты. Это поведение по умолчанию.

Удалить таблицу может только её владелец, владелец схемы или суперпользователь. Чтобы опустошить таблицу, не удаляя её саму, вместо этой команды следует использовать DELETE или TRUNCATE. DROP TABLE всегда удаляет все индексы, правила, триггеры и ограничения, существующие в целевой таблице. Чтобы удалить таблицу, на которую ссылается представление или ограничение внешнего ключа в другой таблице, необходимо дополнительно указать CASCADE. (С указанием CASCADE зависимое представление удаляется полностью, тогда как в случае с ограничением внешнего ключа удаляется именно это ограничение, а не вся таблица, к которой оно относится.)

Задание 1. С помощью скрипта на языке SQL (см. задание 7 в ЛР №6):

- создайте **табличное пространство ts_faculty**, которое должно располагаться в каталоге 'D:/SQL/faculty';
- создайте **базу данных db_faculty_FIO**². База данных должна располагаться в созданном вами табличном пространстве **ts_faculty**;
- создайте в БД «ФАКУЛЬТЕТ» **схему Faculty**. В дальнейшем все объекты БД «ФАКУЛЬТЕТ» необходимо создавать именно в этой схеме.

² В качестве FIO укажите свою фамилию с инициалами или только инициалы

Задание 2. На основе представленной [логической модели БД «Факультет»](#) создайте ее физическую модель на языке SQL, то есть создайте таблицы со всеми необходимыми ограничениями целостности.

- для всех атрибутов задайте наиболее подходящие типы данных ([см. Таблица. Типы данных](#));


- укажите обязательность / необязательность заполнения (значения NULL) полей;
- задайте декларативные ограничения целостности: значения по умолчанию, проверочные ограничения, автозаполнение полей, формулы для генерируемых (вычисляемых) полей.

- определите первичные ключи;
- установите связи (внешние / вторичные ключи), при этом для всех создаваемых связей задайте **каскадное обновление** значений внешнего ключа при изменении объекта ссылки.

2. Использование инструментальных средств с графическим интерфейсом пользователя

Как уже говорилось, в pgAdmin предусмотрены средства для создания БД как с помощью кода на языке SQL, так и с помощью графического интерфейса пользователя. В предыдущих заданиях вы создали БД, таблицы и связи с помощью SQL-скрипта, сейчас рассмотрим, как сделать это с помощью графического интерфейса пользователя. Общий подход при этом – выбор соответствующих опций в контекстном меню базы данных в **Обозревателе объектов pgAdmin**.

2.1. Представление базы данных в виде ER-диаграммы (ERD)

В pgAdmin есть средства для визуального отображения диаграммы базы данных – в виде упрощенного варианта ER-диаграммы. Для ее построения нужно в **Обозревателе объектов pgAdmin** выбрать нужную базу данных и в ее контекстном меню выбрать пункт **Generate ERD**. Свойства таблицы можно открыть с помощью **двойного щелчка** по таблице или с помощью команды  **Edit Table** на панели инструментов или сочетанием клавиш **Alt + Ctrl + E**.

Задание 3. Постройте ER-диаграмму базы данных «ФАКУЛЬТЕТ». Убедитесь, что все требования описанной логической модели соблюдены (таблицы, первичные ключи, связи и их свойства, проверочные ограничения, значения по умолчанию и т.д.).

2.2. Создание и изменение базы данных

В **Обозревателе объектов pgAdmin** в контекстном меню раздела **Базы данных** нужно выбрать опцию **Создать > База данных...**. Откроется окно **Создание База данных**, в котором задаются свойства новой БД. Здесь имеется несколько групп параметров БД, указанных в верхней части окна: **General**, **Определение**, **Безопасность**, **Параметры**, **Дополнительно**, **SQL**.

В группе **General** задается

- имя БД,
- владелец БД – имя пользователя (роли), назначаемого владельцем новой БД, либо DEFAULT, чтобы владельцем стал пользователь по умолчанию (а именно, пользователь, выполняющий команду).
- комментарий.

В группе **Определение** задается еще ряд свойств БД, из которых обратим внимание на следующие:

- *Кодировка* – кодировка символов в новой БД;
- *Шаблон* – имя шаблона, из которого будет создаваться новая БД, либо DEFAULT, чтобы выбрать шаблон по умолчанию (template1);
- *Табличное пространство* – имя табличного пространства, связываемого с новой БД, или DEFAULT для использования табличного пространства шаблона. Это табличное пространство будет использоваться по умолчанию для объектов, создаваемых в этой БД.

После задания всех необходимых свойств, можно с помощью кнопки **Сохранить** запустить создание БД.

Изменить созданную БД можно путем корректировки ее Свойств, которые вызываются через контекстное меню БД в **Обозревателе**.

2.3. Создание и изменение таблицы

Создать новую таблицу можно в **Обозревателе объектов pgAdmin**. Для этого в контекстном меню раздела **Схемы > Таблицы** базы данных вызывается команда **Создать > Таблица....** Откроется окно **Создание Таблица**, в котором можно задать основные параметры таблицы в целом и ее столбцы (поля) с указанием типов и допустимости (или недопустимости) Null-значений. Кроме того, можно задать все **Ограничения** целостности на одноименной вкладке.

2.4. Создание и изменение связей

Создание связей (внешних / вторичных ключей) осуществляется в окне **Свойства Таблица** на вкладке **Ограничения > Внешний ключ**.

Задание 4. Частично заполните созданную БД «ФАКУЛЬТЕТ», внося по 1-2 записи в каждую из таблиц, используя графический интерфейс (команда контекстного меню таблицы **Просмотр/редактирование данных > Все строки**). Предварительно продумайте последовательность заполнения таблиц. Проверьте правильность и корректность работы ограничений, наложенных на атрибуты при создании таблиц.

Задание 5. С помощью SQL-скрипта измените структуру БД «ФАКУЛЬТЕТ». Добавьте новую таблицу Кафедра (Department). Установите связь таблицы Кафедра (Department) с таблицей Преподаватель (Teacher), учитывая, что в штат одной кафедры входит несколько преподавателей, а один преподаватель может совмещать работу на нескольких кафедрах. Продумайте какие атрибуты и в какие таблицы необходимо добавить, а из каких удалить, чтобы реализовать связь указанного типа. Также учтите тот факт, что за кафедрой закреплен только один кабинет (аудитория).

Кафедра (Department)

- *Код кафедры* – обязательный атрибут, первичный ключ;
- *Название кафедры* – обязательный атрибут;
- *Подразделение* – обязательный атрибут, указывает к какому структурному подразделению (институту / факультету) относится кафедра;
- *Кабинет* – обязательный атрибут, указывает, где располагается кафедра;
- *Телефон* – необязательный атрибут, формат ввода '000000', где 0 – любая цифра;
- *Заведующий кафедрой* – необязательный атрибут, содержит код преподавателя, который заведует кафедрой.