**Лабораторные работы по курсу**

**Базы данных**

**Лабораторная работа 4**

**«Создание логической и физической модели базы данных»**

**Москва, 202****4**

Оглавление

[1. Теоретическая часть 3](#_Toc158076071)

[1.1. Проектирование базы данных 3](#_Toc158076072)

[1.2. Логическое моделирование. ER-модель 3](#_Toc158076073)

[1.3. Пример ER-диаграммы 5](#_Toc158076074)

[1.4. Физическое проектирование 7](#_Toc158076075)

[1.4.1. Создание таблиц 7](#_Toc158076076)

[1.4.2. Изменение таблиц 7](#_Toc158076077)

[1.4.3. Удаление таблиц 8](#_Toc158076078)

[1.4.4. Ограничения 8](#_Toc158076079)

[1.5. Работа с программой *pgModeler* 10](#_Toc158076080)

[1.5.1. Подключение к существующей базе данных. 11](#_Toc158076081)

[1.5.2. Импорт базы данных. 12](#_Toc158076082)

[1.5.3. Экспорт базы данных. 12](#_Toc158076083)

[1.5.4. Разработка модели базы данных. 13](#_Toc158076084)

[2. Практическая часть 15](#_Toc158076085)

[2.1. Задание 1. 15](#_Toc158076086)

[2.2. Задание 2. 16](#_Toc158076087)

[2.3. Задание 3. 16](#_Toc158076088)

[Контрольные вопросы 16](#_Toc158076089)

[Список использованной литературы 17](#_Toc158076090)

1. Теоретическая часть
   1. Проектирование базы данных

Проектирование баз данных в общем виде является сложной и трудоемкой задачей. В общем случае её можно сформулировать, как выбор подходящей логической структуры для заданного массива данных, который необходимо поместить в базу данных. [1] Другими словами, необходимо выделить требуемые отношения (таблицы) и содержащиеся в них атрибуты (столбцы). По окончанию проектирования созданную базу данных необходимо преобразовать в форму, которая может быть воспринята конкретной СУБД.

Проектирование базы данных можно разделить на несколько этапов:

* Концептуальное (семантические, инфологическое) – на данном этапе происходит анализ и определение понятий предметной области
* Логическое – создание схемы данных на основе заданной модели без учета специфики СУБД
* Физическое – создание базы данных с учетом специфики СУБД

Данная лабораторная работа посвящена созданию логической и физической моделей базы данных. Логическая модель данных будет изучаться на примере создания ER-модели или модели «*сущность-связь*». Физическая модель данных будет представлена в виде запросов для СУБД PostgreSQL.

* 1. Логическое моделирование. ER-модель

ER-модель является одним из самых распространенных средств для представления структуры баз данных.

В ней структура данных отображается графически в виде диаграммы, которая состоит из элементов трех типов:

* Сущности
* Атрибуты
* Связи

**Сущность** – абстрактный объект определенного типа. Например, в базе данных **Student\_mark** можно выделить сущности *Студент, Преподаватель, Структурное подразделение* и др.

**Атрибут** – свойство множества сущностей. Например, сущности *Студент*, могут быть поставлены в соответствии атрибуты *фамилия*, *имя, номер студенческого билета* и др. У каждой сущности возможно выделить подмножество атрибутов или суммы атрибутов, обладающее свойствами уникальности и неизбыточности – потенциальный ключ. Напомним, что из потенциальных ключей можно выделить один, который будет первичным ключом **(***PK,* от *Primary Key***)**, остальные – альтернативными. Первичный ключ должен однозначно идентифицировать каждую запись в таблице. На каждый атрибут можно наложить ограничения, чтобы не допускать добавление в базу данных некорректных значений.

**Связи** – соединения между двумя и большим числом сущностей. Например, между сущностями *Студент* и *Группы* возможно провести связь – «студенты, обучающиеся в группе». Наиболее распространенными являются бинарные связи, соединяющие два множества сущностей.

Существуют несколько типов связи – 1:1, 1: N, N:N. Рассмотрим их подробнее.

Связь «один к одному» (1:1) – в случае данной связи каждой записи из одной таблицы будет соответствовать одна уникальная запись в другой. Например, у каждого студента есть студенческий билет, поэтому связь между сущностями *Студент* и *Студенческий билет* будет один к одному. Связь между таблицами происходит за счет совпадения значений первичных ключей в обоих таблицах.

Изображение выглядит как текст, Шрифт, число, программное обеспечение

Автоматически созданное описание

Рисунок 1 Связь 1:1

Связь «один ко многим» (1:N) – самый распространенный тип связи. В данном случае, одной записи главной таблицы можно сопоставить несколько записей подчинённой таблицы. Например, в одной группе может обучаться множество студентов. Таким образом, связь между атрибутами сущностей *Группа* и *Студент* будет один ко многим.

Связь между таблицами происходит за счет внешних ключей **(***FK, от Foreign Key***)** Напомним, внешний ключ – поле подчиненной таблицы, соответствующее первичному ключу главной таблицы. Например, атрибут *номер группы* в таблице *Студент* является внешним ключом, так как он связан с первичным ключом *номер группы* таблицы *Группа* Данный атрибут отмечен символом FK.

Изображение выглядит как текст, Шрифт, число, снимок экрана

Автоматически созданное описание

Рисунок 2 Связь 1:N

Связь «Многие ко многим» (N: N) предполагает возможность связи одного или нескольких элементов из одной таблицы с одним или несколькими элементами из другой таблицы. В случае рассматриваемого примера один преподаватель может работать в нескольких структурных подразделениях, а в одном структурном подразделении могут содержаться несколько преподавателей. Данный тип связи нереализуем в рамках физической модели, поэтому представляется в виде создания дополнительной таблицы, содержащей ключевые поля соединяемых отношений. Например, *Структурное подразделение* и *Преподаватель* соединены посредством таблицы *Трудоустройство.*

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 3 Связь N:N

Существует несколько вариантов записи (нотации) ER-моделей в форме диаграмм. В данном лабораторном практикуме будет рассмотрена ER-диаграмма в записи Гордона Эвереста.

Согласно данной нотации, сущность изображается в виде прямоугольника, содержащее её имя. Имя сущности должно быть уникальным в рамках одной модели. Атрибуты сущности записываются внутри данного прямоугольника. Связь изображается линией, которая связывает две сущности, участвующей в отношении. Каждая связь должна именоваться глаголом или глагольной фразой, записываемой над стрелкой. Например, «состоит из», «включает». В зависимости от типа связи возможно различное обозначение конца линии.

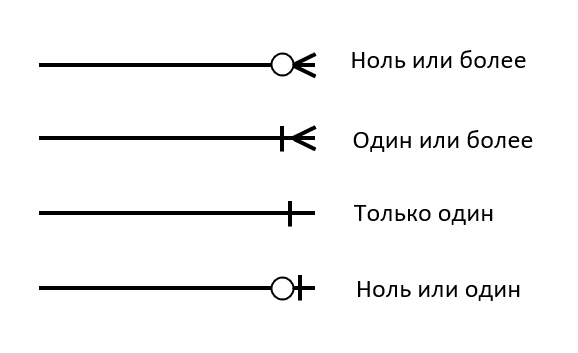


Рисунок 4 Тип линии

В случае связи один к одному, на конце и в начале линии указывается вертикальная черта, в случае один ко многим указывается изображение вилки или вороньей лапки.

* 1. Пример ER-диаграммы

Рассмотрим пример разработки ER-диаграммы на основе базы данных о студентах вуза. В ней необходимо хранить информацию о студенте, группе его обучения, структурном подразделении, в котором состоит эта группа, об оценках за дисциплины и о преподавателях, проставивших эти оценки.

Проанализировав всю требуемую информацию выдели очевидные независимые сущности: *Студент*, *Дисциплина*, *Студенческая группа*, *Преподаватель*, *Структурное подразделение*. Рассмотрим сущности по отдельности.

Сущность ***Студент*** должна содержать всю относящуюся к нему информацию. Выделим следующие атрибуты: *фамилия, имя, отчество, дата рождения*, *номер студенческого билета*, *дата выдачи* *билета*, *дата окончания действия билета,* *номер группы* и *адрес электронной почты.* Номер студенческого билета однозначно определяет студента, то есть обладает свойством уникальности. Поэтому, его можно выбрать в качестве первичного ключа. На практике, часто необходимо на определенные атрибуты накладывать ограничения (*constrain*), позволяющие задавать точный формат записи, диапазон, в котором может лежать значение и др. О данных ограничениях можно задуматься на этапе разработки ER-диаграммы. Добавим ограничение на формат электронной почты. Она будет представлять из себя строку вида [TEXT]@[TEXT].[DOMAIN], где TEXT может содержать любые английские буквы и числа, а DOMAIN только английские буквы.

Обратим внимание, что поля с информацией о студенческом билете (*дату выдачи* *билета*, *дату окончания действия)* относятся к билету, а не к студенту и логично будет разделить данную сущность на две – одна будет описывать студента (человека), вторая – студенческий билет (документ). Исходя из этого выделим еще одну сущность **– *Студенческий билет*** и соединим её с сущностью *Студент* связью 1:1 (каждому студенту будет соответствовать свой студенческий билет). В качестве атрибутов сущности *Студенческий билет* выделим *дату выдачи* *билета* и *дату окончания действия*.

Сущность ***Структурное подразделение*** будет содержать в себе атрибуты *полное и сокращенное название, фамилия руководителя подразделения, телефон*. Для удобства уникальной идентификации введем дополнительный атрибут – *код структурного подразделения*. Он будет являться суррогатным ключом для данной сущности. Напомним, суррогатный ключ – дополнительный атрибут сущности, не имеющий смысловой нагрузки, но являющийся уникальным и неизбыточным, что дает право использовать его в качестве первичного ключа.

Сущность ***Студенческая группа***будет содержать в себе следующие атрибуты: *номер группы* (первичный ключ), *форма обучения* и *код структурного подразделения*, в котором состоит эта группа. На *номер группы* будет наложено ограничение – [ТЕКСТ]-[НОМЕР], где ТЕКСТ – любое буквосочетание на кириллице, а НОМЕР – любые цифры. Значение атрибута *форма обучения* может быть только очной, заочной или очно-заочной. Для обозначения очно-заочной формы к номеру группы добавляется символ «В».

Сущность ***Преподаватель*** будет содержать в себе его *фамилию, имя, отчество, ученую степень, ученое звание, должность, стаж, оклад*. *Ученая степень* должна иметь формат [к/д].[ТЕКСТ].н, где ТЕКСТ – название отрасли наук, а к./д. н. – кандидат/доктор наук. В качестве ключа выбран суррогатный ключ – *код преподавателя*.

Сущность ***Дисциплина***будет включать в себя атрибуты: *код дисциплины, название дисциплины, код структурного подразделения, код преподавателя* (суррогатный ключ*), зачетные единицы трудоемкости (ЗЕТ*), *семестр*.

Создав сущности и их атрибуты, для построения ER-диаграммы осталось определить связи между сущностями.

Сущности *Студент* и *Студенческий билет* можно связать по одноимённому атрибуту - *номер студенческого билета* (связь «один к одному»)*.* *Студенческая группа* и *Студент* связаныпо атрибуту *номер группы* связью «один ко многим». Причем атрибут *номер группы* сущности *Студент* будет внешним ключом. Напомним, внешний ключ – поле подчиненной таблицы, соответствующее первичному ключу главной таблице. Сущности *Структурное подразделение* и *Дисциплина* связаныпо атрибуту *код структурного подразделения* связью «один ко многим». Атрибут *код структурного подразделения* сущности *Дисциплина* является внешним ключом. Сущности *Преподавател*ь и *Дисциплина* также связаны «один ко многим» по атрибуту *код преподавателя* и одноименный атрибут у сущности *Дисциплина* является внешним ключом.

Исходя из того, что один студент может иметь оценки по разным дисциплинам и одной дисциплине могут обучаться несколько студентов, связь между сущностями *Студент* и *Дисциплина* будет «многие ко многим». Для реализации данной связи добавим еще одну сущность - ***Результат освоения дисциплины***, Она будет содержать в себе атрибуты *номер студенческого билета* и *код дисциплины*. Данные атрибуты будут являться внешними ключами для связи с сущностями *Студент* и *Дисциплина.* Также *Результат освоения дисциплины* будет содержать в себе атрибут с оценкой за дисциплину. Допустимые значения атрибута *Оценка* - от 2 до 5 включительно.

Аналогично, связь между сущностями *Преподаватель* и *Структурное подразделение* будет «многие ко многим». Поэтому добавим сущность ***Трудоустройство***, содержащую внешние ключи из связанных таблиц – код структурного подразделения, код преподавателя, а также номер трудового договора и ставку.

Итоговая ER-диаграмма приведена на рисунке 5.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 5 ER-диаграмма студентов

* 1. Физическое проектирование
     1. Создание таблиц

После разработки логической модели возможно создать физическую модель, связанную с конкретной СУБД. В качестве СУБД выбрана PostgreSQL.

Создание таблиц в PostgreSQL происходит с помощью команды CREATE TABLE. Сокращенный синтаксис приведен ниже.

CREATE TABLE [ IF NOT EXISTS ] *имя\_таблицы*

(

имя\_столбца тип\_данных [ *ограничение\_столбца* [ ... ] ]

*[ ограничение\_таблицы ]*

);

Задание ограничений будет подробно рассмотрено в п. 1.4.4.

Пример:

Для создания таблицы *Structural\_units* возможно выполнить следующий SQL запрос.

CREATE TABLE Structural\_units(

    structural\_unit\_id SERIAL,

    unit\_type VARCHAR(64),

    full\_title TEXT,

    abbreviated\_title VARCHAR(20),

    head\_of\_the\_unit VARCHAR(40)

);

* + 1. Изменение таблиц

Для изменения таблиц базы данных используется команда ALTER TABLE.

ALTER TABLE [ IF EXISTS ] [ ONLY ] *имя* [ \* ]

*действие* [, ... ]

В качестве действия могут быть добавление, изменение и удаление конкретного столбца, изменение типов данных и др. Синтаксис некоторых возможных действий представлен ниже:

ALTER TABLE [ IF EXISTS ] [ ONLY ] *имя* [ \* ]

ADD [ COLUMN ] [ IF NOT EXISTS ] *имя\_столбца* *тип\_данных* [ COLLATE *правило\_сортировки* ] [ *ограничение\_столбца* [ ... ] ]

DROP [ COLUMN ] [ IF EXISTS ] *имя\_столбца* [ RESTRICT | CASCADE ]

ALTER [ COLUMN ] *имя\_столбца* [ SET DATA ] TYPE *тип\_данных* [ COLLATE *правило\_сортировки* ] [ USING *выражение* ]

ALTER [ COLUMN ] *имя\_столбца* SET DEFAULT *выражение*

ALTER [ COLUMN ] *имя\_столбца* DROP DEFAULT

Значение всех параметров команды можно посмотреть в литературе [2], В данной лабораторной работе будут использоваться параметры, приведенные в примерах ниже. Примеры:

Добавление в таблицу *Students* столбца *address*:

ALTER TABLE student

ADD COLUMN address varchar(30);

Изменение столбца адрес – увеличение его объема.

ALTER TABLE student

ALTER COLUMN address TYPE varchar(80);

Удаление столбца с адресом

ALTER TABLE student

DROP COLUMN address;

* + 1. Удаление таблиц

Удаление таблицы происходит с помощью команды DROP TABLE.

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

Параметры:

IF EXISTS – не считать ошибкой, если таблица не существует, выдается только замечание.

CASCADE - автоматически удаляются объекты, зависящие от данной таблицы (например, представления).

RESTRICT - отказ в удалении таблицы, если от неё зависят какие-либо объекты (по умолчанию).

Пример удаления таблицы *Students*

DROP TABLE students;

* + 1. Ограничения

Отнесение данных к определенному типу (например, INTEGER, DATE) накладывает на них общие ограничения. Однако, часто требуются, чтобы данные находились в необходимом диапазоне или имели строго определенный вид. Например, значение атрибута *оценка* должно быть целым и лежать в диапазоне от 2 до 5. Для указания таких ограничений существуют специальные ключевые слова.

Таблица 2 Ключевые слова для установки ограничений

|  |  |
| --- | --- |
| Ключевое слово | Описание |
| PRIMARY KEY | Первичный ключ |
| FOREIGN KEY…REFERENCES | Внешний ключ |
| NOT NULL | Значение не может принимать значение NULL |
| NULL | Значение может принимать значение NULL |
| CHECK (условие) | Проверка условия |
| UNIQUE | Уникальность атрибута |
| DEFAULT | Установка значения по умолчанию |

Самый простой способ установки ограничений – использование ключевого слова CHECK. В его определении возможно указать, что значение данного столбца будет удовлетворять логическому выражению (проверке истинности).

Например, укажем, что оценка может быть от 2 до 5:

mark INTEGER NOT NULL CHECK (mark >=2 AND mark <=5)

Ограничения NOT NULL/ NULL указывает на то, что столбцу нельзя /можно присваивать значение NULL.

Ограничение уникальности UNIQUE, гарантирует, что данные в столбце должны быть уникальными для всех строк таблицы.

Например:

patronymic VARCHAR(30) NULL

birthday DATE NOT NULL

email VARCHAR(30) UNIQUE

Если NOT NULL и UNIQUE используют вместе, то это ограничение эквивалентно определению первичного ключа. Однако, ограничений уникальности и NOT NULL может быть сколько угодно, а первичный ключ может быть только один.

Для указания того, что атрибут является первичным ключом, используются ключевые слова PRIMARY KEY. Например,

structural\_unit\_id SERIAL PRIMARY KEY

Ограничение FOREIGN KEY, подразумевает, что данный столбец является внешним ключом, то есть его значения должны быть связаны со значениями аналогичного столбца в другой таблицы, являющейся главной. После слова FOREIGN KEY в скобках указывают названия столбцов подчиненной таблицы, которые являются внешними ключами. После слова REFERENCES указывают название главной таблицы, а в скобках ее атрибут, по которому эти таблицы связаны. В случае если внешний ключ один, слово FOREIGN KEY можно опустить.

Например, для указания ссылки на таблицу со структурными подразделениями из таблицы студенческих групп можно выполнить следующим образом:

structural\_unit\_id INTEGER NOT NULL REFERENCES Structural\_units(structural\_unit\_id)

Аналогично возможно выполнить ограничение с помощью FOREIGN KEY

FOREIGN KEY(structural\_unit\_id)

            REFERENCES Structural\_units(structural\_unit\_id)

Параметр DEFAULT <выражение> задает значение для столбца, которое будет присваиваться ему по умолчанию. Тип выражения должен соответствовать типу столбца. Это выражение будет присвоено столбцу во всех операциях добавления данных, где его значение не задается напрямую. В случае, если параметр DEFAULT не используется, значением по умолчанию будет NULL.

Каждому ограничению возможно присвоить отдельное имя. Для этого перед ним указывается ключевое слово **CONSTRAINT и название ограничения после него.**

CONSTRAINT email\_cheak

        CHECK (email ~\* '^[A-Za-z0-9.\_+%-]+@[A-Za-z0-9.-]+[.][A-Za-z]+$')

Все ограничения возможно выносить за пределы строки с атрибутом.

Рассмотрим пример создания таблицы «Студент»

CREATE TABLE Student(

    student\_id INTEGER NOT NULL,

    surname VARCHAR(30) NOT NULL,

    name VARCHAR(30) NOT NULL,

    patronymic VARCHAR(30) NULL,

    students\_group\_number VARCHAR(7) NOT NULL,

    birthday DATE NOT NULL,

    email VARCHAR(30) UNIQUE,

    PRIMARY KEY(student\_id),

    CONSTRAINT Students\_group\_key

        FOREIGN KEY(students\_group\_number)

            REFERENCES Students\_group(students\_group\_number)

            ON DELETE CASCADE,

    CONSTRAINT email\_cheak

        CHECK (email ~\* '^[A-Za-z0-9.\_+%-]+@[A-Za-z0-9.-]+[.][A-Za-z]+$')

);

В данном случае все ограничения на ключи и CHECK были вынесены за пределы строк с атрибутами. Для указания внешнего ключа использовалось ключевое слово FOREIGN KEY. Для обеспечения каскадного удаления всех связанных с записью строк по ключам используются ключевые слова ON DELETE CASCADE.

В завершении работы приведем итоговую схему данных с учетом всех заданных типов.

Изображение выглядит как текст, Шрифт, линия, число

Автоматически созданное описание­

Рисунок 6 Схема данных

На практике использование кириллических символов при разработке физической модели не используется, поэтому учебная база данных полностью написана латинскими буквами.

* 1. Работа с программой pgModeler

ПриложениеpgModeler **–** open-source проект, предназначенный для моделирования баз данных. Программа позволяет создавать графические модели базы данный и экспортировать их в файлы различных форматов – от SQL-скриптов для создания физической модели, до изображения имеющих тип png.

Для запуска программы необходимо в командной строке набрать название программы – *pgmodeler*. Окно запустившейся программы выглядит следующим образом:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 7 Основное окно pgModuler

Для создания нового проекта необходимо нажать на кнопку «New model» или на рабочей панели сверху выбрать пункт с символом  . Аналогичного действия возможно добиться, нажав комбинацию клавиш Ctrl+N.

После создания новой модели откроется рабочее окно, на котором будет производится разработка модели. Рассмотрим кратко основные функции *pgmodeler*.

1. Основные элементы управления.

Элементы управления располагаются в левой части экрана.

|  |  |
| --- | --- |
|  | Окно приветствие пользователей. Содержит базовый функционал для создания и открытия проекта. |
| Изображение выглядит как текст, снимок экрана, Прямоугольник, логотип  Автоматически созданное описание | Окно разработки даталогической модели. |
| Изображение выглядит как текст, снимок экрана, логотип, символ  Автоматически созданное описание | Окно для подключения к существующей базе данных. |
| Изображение выглядит как текст, снимок экрана, логотип, Шрифт  Автоматически созданное описание | Окно для импорта модели из существующей базы данных в программу *pgmodeler* |
| Изображение выглядит как текст, логотип, снимок экрана, Шрифт  Автоматически созданное описание | Окно для экспорта модели на сервер баз данных, в формат SQL файла, графический формат или в html. |
| Изображение выглядит как снимок экрана, текст, аптечка первой помощи  Автоматически созданное описание | Окно для сравнения нескольких моделей и определения отличий между ними |
| Изображение выглядит как текст, символ, Шрифт, дизайн  Автоматически созданное описание | Окно настроек программы |

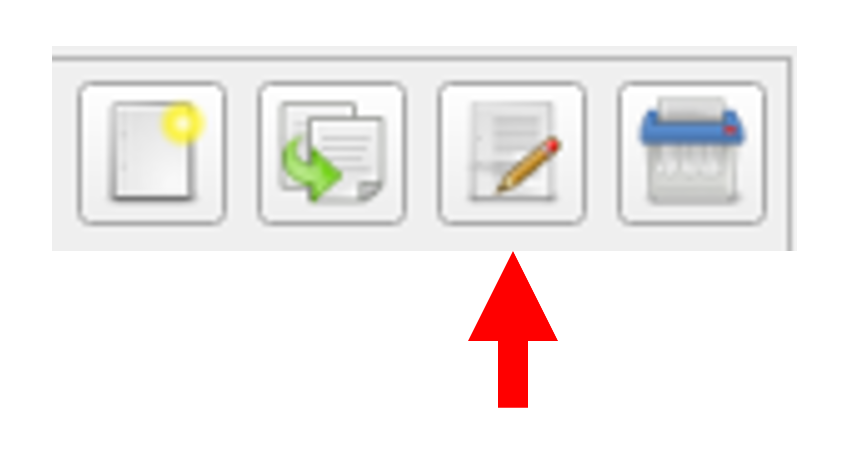
* + 1. Подключение к существующей базе данных.

Для подключения к существующей базе данных в поле Manage необходимо в выпадающем слева сверху окне найти и отредактировать существующее стандартное подключение или добавить новое.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

При выборе «Edit connections», выберете кнопку “Edit selected connection”.



В открывшейся вкладке необходимо указать значения, соответствующие вашему соединению. Их вы можете посмотреть в настройках подключения в программе *pgadmin*.

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

После настройки подключения, проведите его тестирование, нажав на кнопку Test. В случае удачного соединения будет выведено окно с сообщением Success.

Изображение выглядит как текст, снимок экрана, Шрифт, логотип

Автоматически созданное описание

* + 1. Импорт базы данных.

При успешном подключении к серверу возможно импортировать схему базы данных в ваш проект. Для этого в окне import нужно выбрать соединение с нашим сервером и в правом выпадающем окне выбрать необходимую базу данных.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

После импорта в ваш проект будет добавлена ERD модель базы данных.

* + 1. Экспорт базы данных.

Для экспорта проекта в окне *Export* необходимо выбрать желаемый выходной формат файла – sql, png или html. Далее укажите путь, по которому файл должен быть сохранен и нажмите кнопку *Export* в нижнем углу открывшегося экрана.

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

* + 1. Разработка модели базы данных.

В качестве примера рассмотрим добавление в импортированную базу данных сущности «Парковочное место», связанное с преподавателем. Для добавления новой таблицы щелкнете правой кнопкой мыши по рабочему пространству окна *design* и в выпадающем окне выберете вкладку New -> Table

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

Заполним открывшееся окно данными. В поле Name внесем название таблицы – car\_places.

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

В нижнем поле, во вкладке Columns добавим атрибуты созданной сущности. Для этого нажмем на кнопку «Add Item».

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

В открывшемся окне укажем название атрибута – place\_id. Тип – smallint. Аналогично создадим столбец owner целочисленного типа integer, car\_number, состоящий из 8 символов и логическое поле valid, по умолчанию хранящее в себе значение TRUE.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

В качестве первичного ключа выберем значение place\_id. Для этого установим галочку в левой колонке PK. Нажав кнопку *Apply*, создадим таблицу.

Добавим ограничения на столбцы для созданной таблицы. Предположим, что владельцем машины может быть только кто-то из профессоров. Для этого создадим внешний ключ и соединим его с таблицей *Professors*. Для этого дважды щелкнем по созданной таблице и перейдем во вкладку Constraints. По кнопке «Add Item» войдем в окно добавления ограничений. Далее выставим настройки следующим образом: Name – owner\_fk, Contraint Type – FORIENG KEY, Match – MATCH SIMPLE. Далее в поле Columns выберем столбец owner и добавим его с помощью кнопки «Add Item». Аналогично в поле Reference columns выберем таблицу *Professors* и поле professor\_id.

Подтвердите настройки, нажав кнопку Apply.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Для атрибута car\_number добавим проверку на корректность введенного номера автомобиля. Создадим ограничение car\_min\_valid и выберем Constraint type CHECH. В поле ниже укажем значение регулярного выражения для российских автомобильных номеров - car\_number ~\* '^[АВЕКМНОРСТУХ]\d{3}(?<!000)[АВЕКМНОРСТУХ]{2}\d{2,3}$'.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

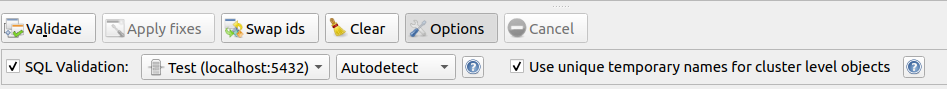
Автоматически созданное описание

Подтвердим создание ограничения и вернемся в основное рабочее окно. Созданная таблица получила связь с таблицей преподавателей.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Перед тем, как экспортировать измененную модель обратно на сервер, необходимо убедиться в том, что после внесенных изменений не были нарушены ограничения, связанные с базой данных. Например, не создана таблица, имеющая аналогичное название на сервере. Для этого в нижней вкладке необходимо выбрать пункт Validation. Установим значение SQL Validation, выберем адрес сервера и нажмем кнопку Validate.



Если валидация продет успешно, то появится сообщение «Database model successfully validated». Далее, экспортируем модель обратно на сервер. Для этого перейдем во вкладку Export, выберем сервер базы данных и установим пункт Ignore object duplicity.

Изображение выглядит как текст, Шрифт, снимок экрана, линия

Автоматически созданное описание

Нажмем на кнопку Export и обновим базу данных на сервере. Вернувшись в pgAdmin, и выполнив запрос, убедимся в том, что таблица была создана.

INSERT INTO car\_places VALUES (1,81001,'А777ТМ77')

SELECT \* FROM car\_places

place\_id | owner | car\_number | valid

----------+-------+------------+-------

1 | 81001 | А777ТМ77 | t

1. Практическая часть

Вариант выбирается в соответствии с формулой: N = (Nв списке mod 10) + 1.

* 1. Задание 1.
     1. В учебной базе данных одним из допущений является возможность прикрепить только одного преподавателя к дисциплине. Исправьте его.
  2. Задание 2.

Отредактируйте базу данных в соответствии с вашим вариантом.

|  |  |
| --- | --- |
| Вариант | Описание |
| 1 | Добавьте в таблицу *student\_id* поле, содержащее цвет студенческого билета. Цвет может быть синим, белым или зеленым. |
| 2 | Добавьте в таблицу *student\_id* поле, содержащее статус студенческого билета: «заблокирован», «активен». |
| 3 | Добавьте в таблицу *student* поле «Пол», способное принимать значения «м» и «ж» |
| 4 | Добавьте в таблицу *student* поле «Паспорт», способное хранить значения в формате XX-XXXXXX, где X – цифра. |
| 5 | Добавьте в таблицу *student* поле «СНИЛС», способное хранить значения в формате XXX-XXX-XXX X, где X – цифра. |
| 6 | Добавьте в таблицу student поле «ИНН», способное хранить значения в формате XXNNAAAAAABB, где все указанные символы – цифры. XX – код субъекта РФ, может быть выбран из любых 5 субъектов, на ваш выбор. |
| 7 | Добавьте в таблицу *professor* поле, содержащее его контактный телефон. Сделайте ограничение, позволяющее хранить номер телефона в формате: +7(XXX)XXX-XX-XX |
| 8 | Добавьте в таблицу *student* поле, содержащее его контактный телефон. Сделайте ограничение, позволяющее хранить номер телефона в формате: 8(XXX)XXX-XX-XX |
| 9 | Добавьте в таблицу *structural\_unit* поле, содержащее контактный телефон подразделения. Сделайте ограничение, позволяющее хранить номер телефона в формате: AA-XX, где AA могут принимать значения 29, 25, 14, 33, 87. |
| 10 | Добавьте в таблицу *students\_group* поле, содержащее данные о кураторе группы. В его качестве может выступать один из преподавателей вуза. |

* 1. Задание 3.
     1. В соответствии с вариантом доработайте логическую модель базы данных. При доработке БД должно быть добавлено не менее трех новых таблиц. Постройте схему новой базы данных в редакторе *pgmodeler* или *Erwin.* Экспортируйте её в созданную базу данных.

|  |  |
| --- | --- |
| Вариант | Описание |
| 1 | Добавить возможность прикрепления студентов на места практики |
| 2 | Добавить возможность занятости студентов в спортивных секциях |
| 3 | Добавить студенческие объединения |
| 4 | Добавить военную кафедру |
| 5 | Добавить общежитие |
| 6 | Добавить медпункт и возможность выхода студента на больничный |
| 7 | Добавить возможность получения студентами льгот и стипендии |
| 8 | Добавить курсы повышения квалификации для преподавателей |
| 9 | Добавить портфолио студентам |
| 10 | Добавить студенческий офис |

Контрольные вопросы

1. Для чего необходимо проектирование базы данных?
2. Что такое ER-модель?
3. Перечислите известные вам типы связи?
4. Какие образом возможно физически реализовать связь «многие ко многим»?
5. Какие команды для установки ограничения вам известны?

Список использованной литературы

|  |  |
| --- | --- |
| [1] | К. Дейт, Введение в системы баз данных, Москва: Диалектика, 2019, p. 1316. |
| [2] | Документация к PostgreSQL 15.1, 2022. |