МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра программной инженерии

УТВЕРЖДАЮ:
проректор по учебной работе
Локтионова О.Г.
«__» _____ 2023 г.

Разработка CRUD-приложения

Методические указания к лабораторным работам
по дисциплине «Базы данных»
для студентов направления подготовки
09.03.04 «Программная инженерия»

Курск 2023

УДК 004.65

Составитель Е.И.Аникина

Рецензент
Кандидат технических наук, доцент кафедры программной инженерии *Е.А.Петрик*

Разработка CRUD-приложения: методические указания к лабораторным работам по дисциплине «Базы данных» для студентов направления подготовки 09.03.04 «Программная инженерия»/Юго-Зап. гос. ун-т; сост. Е.И. Аникина. Курск, 2023. 15 с.

Содержит задания к лабораторным работам, теоретические сведения и примеры решения задач по теме курса, связанной с разработкой интерфейса пользователя и реализацией функционала приложения для работы с базой данных.

Предназначено для студентов направления подготовки бакалавров 09.03.04 «Программная инженерия».

Текст печатается в авторской редакции.

Подписано в печать . Формат 60x84 1/16.
Усл. печ. л. . Уч.-изд. л. . Тираж 100 экз. Заказ . Бесплатно.

Юго-Западный государственный университет
305040, Курск, ул.50 лет Октября, 94.

Лабораторная работа №8

CRUD-ПРИЛОЖЕНИЕ. РАЗРАБОТКА ИНТЕРФЕЙСА ПОЛЬЗОВАТЕЛЯ

ЗАДАНИЕ

Для базы данных, созданной при выполнении лабораторной работы №6, требуется разработать приложение, с помощью которого пользователь может поддерживать базу данных в актуальном состоянии.

Приложение должно иметь многооконный *интерфейс , включающий в себя* главную форму и по отдельной форме для работы с каждой из таблиц вашей БД.

При запуске приложения должна появляться ***главная форма***, в заголовке которой должно быть указано название предметной области, для которой оно создано . На форме должно быть размещено меню (или командные кнопки) для перехода к формам для работы с таблицами БД.

Предлагается для удобства пользователей организовать ввод новых данных через специально разработанные для каждой из таблиц формы.

С помощью ***форм для каждой из таблиц БД*** пользователь должен иметь возможность выполнять следующие действия:

1. Просматривать таблицу с данными (назания полей таблицы должны выводиться на русском языке).
2. Добавлять новые строки таблицы
3. Удалять ставшие ненужными строки таблицы
4. Редактировать содержимое строк
5. Сохранять изменения в базе данных.

Краткие теоретические сведения

Приложение, которое мы начали разрабатывать - это типовое ***CRUD-приложение.***
Аббревиатура ***CRUD*** *означает набор базовых операций с базой данных:*
C*reate*
R*etrieve/****R****ead*
U*pdate*
D*elete*

Операции добавления, удаления и редактирования предназначены для поддержания БД в актуальном состоянии.

Актуальное состояние БД **означает,** что хранящиеся в ней данные соответствуют реальному положению дел в предметной области в данный момент времени.

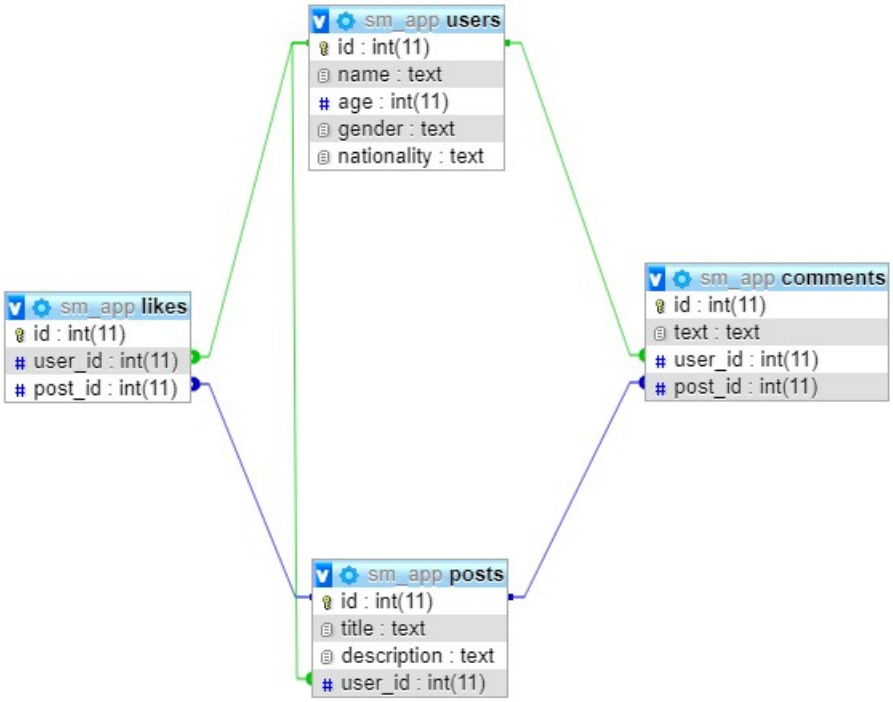
Работа с базой данных SQLite на Python

1. Пример Схемы базы данных для обучения

Разработаем очень маленькую базу данных приложения для социальных сетей. База данных будет состоять из четырех таблиц:

- users
- posts
- comments
- likes

Схема базы данных показана на рисунке ниже.



Пользователи (users) и публикации (posts) будут находиться иметь тип связи один-ко-многим: одному читателю может понравиться несколько постов.

Точно так же один и тот же юзер может оставлять много комментариев (comments), а один пост может иметь несколько комментариев. Таким образом, и users, и posts по отношению к comments имеют тот же тип связи. А лайки (likes) в этом плане идентичны комментариям.

2. Подключение к базе данных

Прежде чем взаимодействовать с любой базой данных через SQL-библиотеку, с ней необходимо связаться.

SQLite, вероятно, является самой простой базой данных, к которой можно подключиться с помощью Python, поскольку для этого не требуется устанавливать какие-либо внешние модули. По умолчанию стандартная библиотека Python уже содержит модуль sqlite3.

Более того, SQLite база данных не требует сервера и самодостаточна, то есть просто читает и записывает данные в файл. Подключимся с помощью sqlite3 к базе данных:

```
import sqlite3
from sqlite3 import Error

def create_connection(path):
    connection = None
    try:
        connection = sqlite3.connect(path)
        print("Connection to SQLite DB successful")
    except Error as e:
        print(f"The error '{e}' occurred")

    return connection
```

Вот как работает этот код:

- Строки 1 и 2 – импорт sqlite3 и класса Error.
- Строка 4 определяет функцию create_connection(), которая принимает путь к базе данных SQLite.

- Строка 7 использует метод connect() и принимает в качестве параметра путь к базе данных SQLite. Если база данных в указанном месте существует, будет установлено соединение. В противном случае по указанному пути будет создана новая база данных и так же установлено соединение.
- В строке 8 выводится состояние успешного подключения к базе данных.
- Строка 9 перехватывает любое исключение, которое может быть получено, если методу .connect() не удастся установить соединение.
- В строке 10 отображается сообщение об ошибке в консоли.

sqlite3.connect(path) возвращает объект connection. Этот объект может использоваться для выполнения запросов к базе данных SQLite. Следующий скрипт формирует соединение с базой данных SQLite:

```
connection = create_connection("E:\\sm_app.sqlite")
```

Выполнив вышеуказанный скрипт, вы увидите, как в корневом каталоге диска E появится файл базы данных sm_app.sqlite. Конечно, вы можете изменить местоположение в соответствии с вашими интересами.

3. Создание таблиц

В предыдущем разделе мы увидели, как подключаться к серверу баз данных SQLite, используя разные библиотеку Python. Мы создали базу данных sm_app . В данном разделе мы рассмотрим, как формировать таблицы внутри баз данных.

Как обсуждалось ранее, нам нужно получить и связать четыре таблицы:

- users
- posts
- comments

- likes

Для выполнения запросов в SQLite используется метод cursor.execute(). В этом разделе мы определим функцию execute_query(), которая использует этот метод. Функция будет принимать объект connection и строку запроса. Далее строка запроса будет передаваться методу execute( ). В этом разделе он будет использоваться для формирования таблиц, а в следующих – мы применим его для выполнения запросов на обновление и удаление.

Примечание. Описываемый далее скрипт – часть того же файла, в котором мы описали соединение с базой данных SQLite.

Итак, начнем с определения функции execute_query():

```
    def execute_query(connection, query):
    cursor = connection.cursor()
    try:
        cursor.execute(query)
        connection.commit()
        print("Query executed successfully")
    except Error as e:
        print(f"The error '{e}' occurred")
```

Теперь напишем передаваемый запрос (query):

```
create_users_table = """
CREATE TABLE IF NOT EXISTS users (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  name TEXT NOT NULL,
  age INTEGER,
  gender TEXT,
  nationality TEXT
);
"""
```

В запросе говорится, что нужно создать таблицу users со следующими пятью столбцами:

- id
- name

- age
- gender
- nationality

Наконец, чтобы появилась таблица, вызываем execute_query(). Передаём объект connection, который мы описали в предыдущем разделе, вместе с только что подготовленной строкой запроса create_users_table:

```
execute_query(connection, create_users_table)
```

Следующий запрос используется для создания таблицы posts:

```
create_posts_table = """
CREATE TABLE IF NOT EXISTS posts(
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  title TEXT NOT NULL,
  description TEXT NOT NULL,
  user_id INTEGER NOT NULL,
  FOREIGN KEY (user_id) REFERENCES users (id)
);
"""
```

Поскольку между users и posts имеет место отношение один-ко-многим, в таблице появляется ключ user_id, который ссылается на столбец id в таблице users. Выполняем следующий скрипт для построения таблицы posts:

```
execute_query(connection, create_posts_table)
```

Для работы пользователей с созданной базой данных было разработано десктопное приложение. При разработке приложения использовался язык Python и библиотека PySimpleGUI.

Приложение представляет собой набор взаимосвязанных форм.

Структура приложения показана на рисунках 1-6.

На рисунке 1 представлено окно входа в приложение. В приложении работают ограничения на вход согласно данным пользователя.

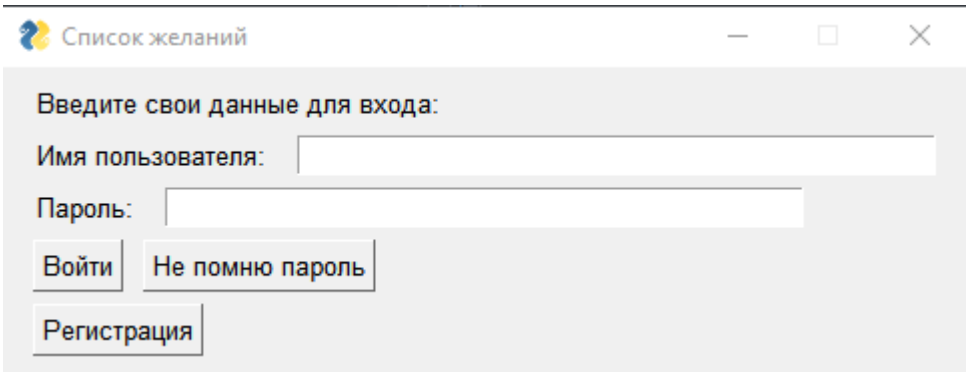


Рисунок 1 – Домашняя страница

На рисунке 2 представлено окно «Регистрация пользователя».

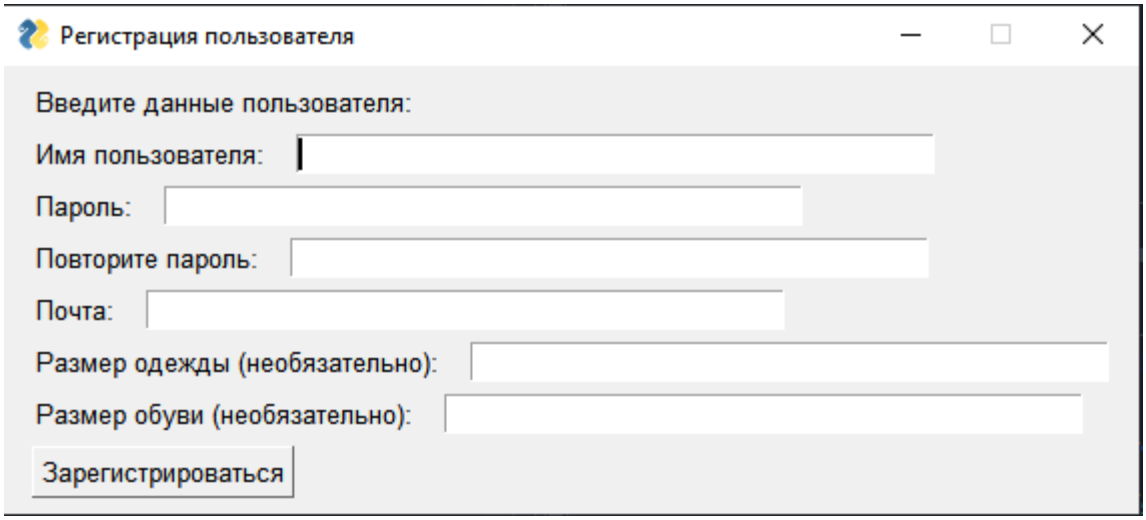


Рисунок 2 – Окно «Регистрация пользователя»

На рисунке 3 представлено окно «Список желаний пользователя».

Рисунок 3 – Окно «Список желаний»

На рисунке 4 представлено окно «Элемент списка желаний».

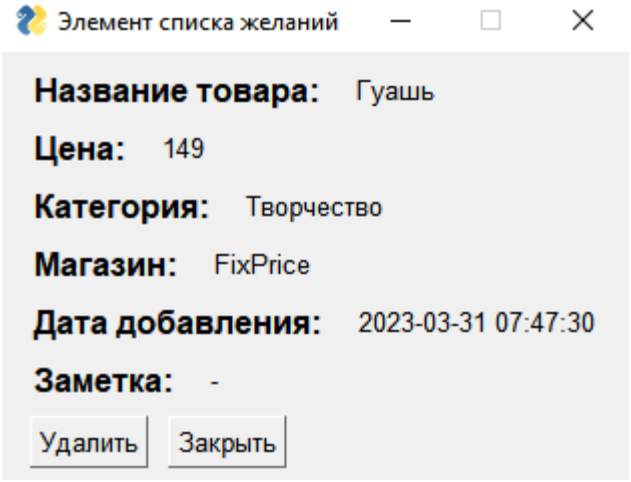


Рисунок 4 – Окно «Элемент списка желаний»

На рисунке 5 представлено окно «Выбор товара для добавления в список желаний».

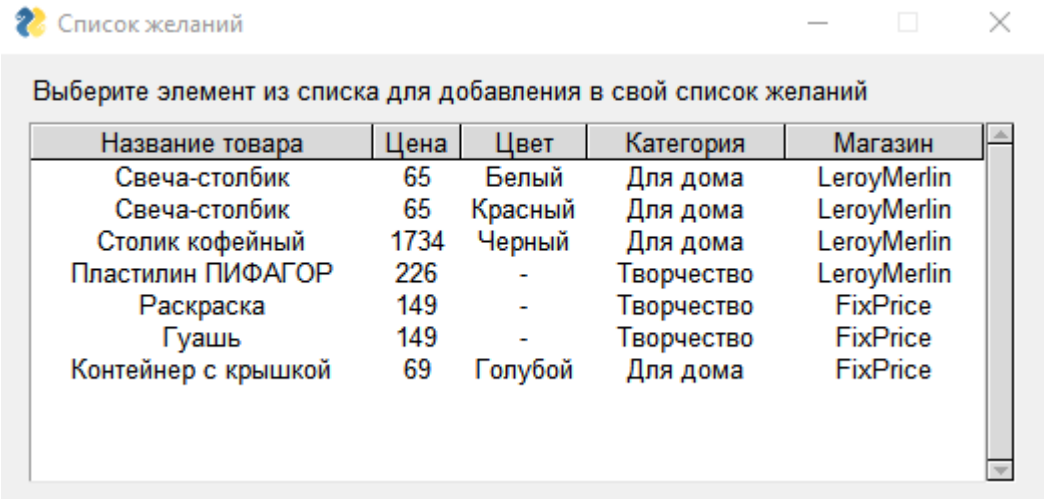


Рисунок 5 – Окно «Выбор товара для добавления в список желаний»

На рисунке 6 представлено окно «Добавление товара в список желаний».

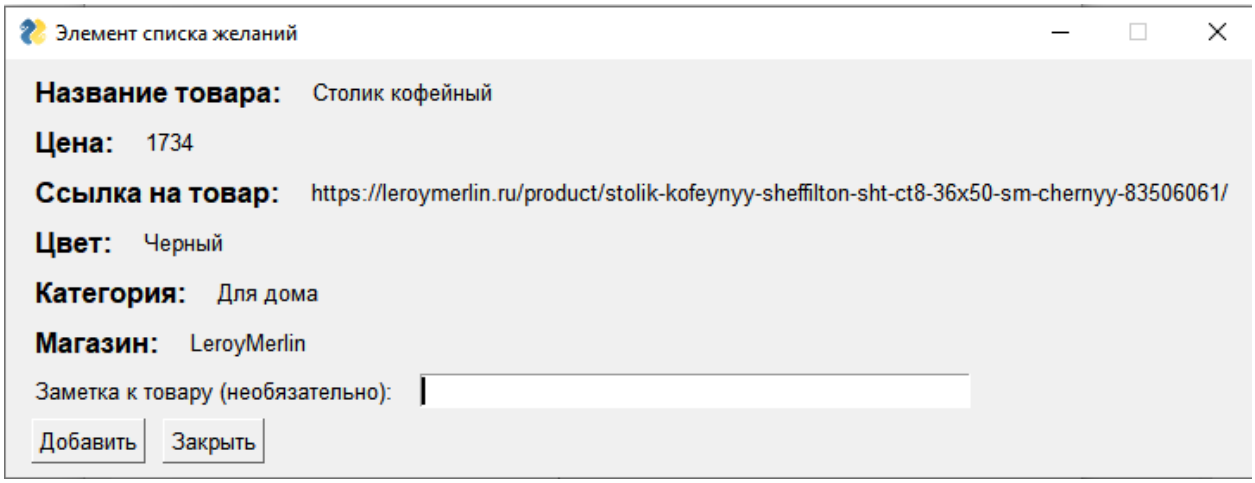


Рисунок 6 – Окно «Добавление товара в список желаний»

ЛАБОРАТОРНАЯ РАБОТА №9

CRUD-ПРИЛОЖЕНИЕ

РАЗРАБОТКА И ПРОГРАММНАЯ РЕАЛИЗАЦИЯ СИСТЕМЫ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ

ЗАДАНИЕ

1. Разработайте для вашей БД набор ограничений доменной, целостности данных, программная реализация которых должна гарантировать, что никакие действия с записями таблиц не приведут к аномалиям в данных вашей базы.

2. Для каждой из таблиц вашей БД надо разработать алгоритмы для добавления и редактирования строк с обязательной проверкой значений всех полей на допустимый тип данных и соответствие ограничениям на допустимые значения данных в соответствии с разработанными ограничениями доменной целостности

3. Разработайте алгоритмы каскадного удаления строк из связанных таблиц.

4. Модернизируйте приложение, разработанное вами в лабораторной работе №8 , дополнив его программной реализацией разработанных ограничений доменной, целостности данных

ТЕОРИЯ

Ограничения целостности можно определить как специальные средства в базах данных, главное назначение которых - не дать попасть в базу недопустимым данным (например, предупредить ошибки пользователей при вводе данных).

Целостность базы данных (англ. database integrity) — соответствие имеющейся в базе данных информации ее внутренней логике, структуре и всем явно заданным правилам. Каждое правило, налагающее некоторое ограничение на возможное состояние базы данных, называется ограничением целостности.

Примеры правил: вес детали должен быть положительным; количество знаков в телефонном номере не должно превышать 15; возраст родителей не может быть меньше возраста их биологического ребенка и так далее.

Понятие целостности данных является ключевым понятием баз данных.

Все ограничения целостности можно разделить на три большие категории:

· первая категория - ***доменная целостность.*** Они отвечают за то, чтобы в соответствующем поле базы данных были допустимые значения. Например, фамилия, как правило, должна состоять из букв, а почтовый индекс - из цифр. В базах данных такая целостность обычно обеспечивается условиями на значение, запретом пустых значений, триггерами и хранимыми процедурами, а также ключами;

· вторая категория - ***сущностная целостность*** . Главная задача здесь - сделать так, чтобы данные об одной сущности не попали в базу данных два раза. Обеспечивается ограничением уникальности и первичным ключом;

· третья категория - ***ссылочная целостность*** , обеспечивается системой первичных и внешних ключей. Например, при помощи этих средств можно гарантировать, что у нас не будет заказов, оформленных на покупателей, которых нет в базе данных.

Надо сказать, что наличие развитой системы ограничений целостности во многом определяет зрелость базы данных. Обычно проще сразу позаботиться о том, чтобы в базу данных не попадали неверные значения, чем потом их убирать из базы данных.

Кроме того, при создании ограничений целостности разработчики должны позаботиться о том, чтобы ошибки, возникающие при нарушениях целостности, перехватывались клиентским приложением.

Обеспечение целостности данных является важнейшей задачей при проектировании и эксплуатации систем обработки данных (СОД).

Целостность является одним из аспектов информационной безопасности наряду с доступностью - возможностью с приемлемыми затратами получить требуемую информационную услугу, и конфиденциальностью - защитой от несанкционированного прочтения.

При выполнении операций над БД проверяется выполнение ограничений целостности. Действия, приводящие к нарушению подобных ограничений, отвергаются.

Работу системы по проверке ограничений можно изобразить на следующем рисунке:

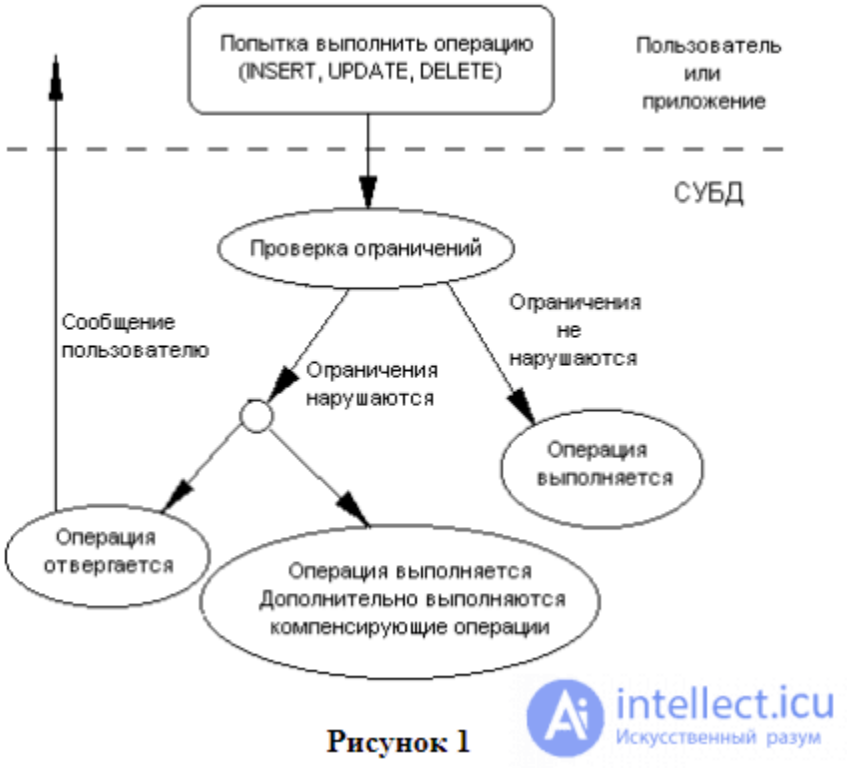


Рисунок 1

Рисунок 1

Пример определения набора ограничений целостности для БД «Деканат»

Фрагмент базы данных для деканата состоит из 3-х таблиц: Студенты, Дисциплины и Результаты экзаменов.

Для каждой из таблиц надо разработать:

1) правила проверки значений во всех столбцах на соответствие определенному типу данных и соответствие ограничений на допустимые значения данных,

2) правила для добавления строк

3) правила для удаления строк.

Таблица **Студенты** (данные о множестве объектов)

Номер_зачетной _книжки (PK)	Фамилия, имя, отчество	Дата рождения	Домашний адрес

Таблица **Дисциплины** (данные о множестве объектов)

Код дисциплины (PK)	Название дисциплины	Объем в часах

Таблица **Результаты экзаменов** (данные о множестве связей между объектами Студенты и Дисциплины)

ID результата (PK)	Код дисциплины (FK)	Номер_зачетной_книжки (FK)	Дата экзамена	Оценка

Ограничения целостности для таблицы *СТУДЕНТЫ*

6. ***Номер_зачетной _книжки*** является уникальным для каждого студента .
7. ***Номер_зачетной _книжки –*** строка из 6-ти символов, разрешены только цифры. Первая цифра не может быть равна 0.

3. ***Фамилия, имя, отчество –*** строка символов, длиной до 50 символов. Может содержать только буквы русского алфавита

4. ***Дата рождения*** – календарная дата .Предположим, что студенты не могут быть моложе 16-ти лет и старше 60-ти лет (сообразите, в каком диапазоне могут находиться даты рождения).

5. При удалении данных о СТУДЕНТЕ необходимо удалить из таблицы Результаты_экзаменов все записи с таким же значением в поле ***Номер зачетной книжки***.

Ограничения целостности для таблицы *Дисциплины*

Правила для контроля уникальности в ключевом поле и требования к типам данных и ограничения на допустимые значения данных во всех полях разрабатываются по аналогии с приведенными для таблицы Студенты.

При удалении данных о Дисциплине необходимо удалить из таблицы Результаты_экзаменов все записи с таким же значением в поле Код дисциплины.

Ограничения целостности для таблицы *Результаты экзаменов*

Правила для контроля уникальности в поле первичного ключа ID и требования к типам данных и ограничения на допустимые значения данных во всех полях разрабатываются по аналогии с приведенными для таблицы Студенты.

Результаты экзаменов можно добавить только для уже существующих в базе Студентов и Дисциплин.