

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет Информатика и вычислительная техника

(наименование факультета)

Кафедра Программное обеспечение вычислительной техники и автоматизированных систем

(наименование кафедры)

|  |  |  |
| --- | --- | --- |
|  | | |
| Зав. кафедрой | | «ПОВТиАС» |
| \_\_\_\_\_\_\_\_\_\_\_\_ | | В.В. Долгов |
| (подпись) | | (И.О.Ф.) |
| «\_\_\_» | \_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г. | |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе по дисциплине (модулю) Базы данных\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(наименование учебной дисциплины (модуля))

на тему: Проектирование и реализация базы данных «Библиотека»

Автор проекта (работы) М.Л. Лапардин

(подпись) И.О.Ф.

Направление/специальность, профиль/специализация:

09.03.04 Программная инженерия

код направления наименование направления (специальности)

Программное обеспечение вычислительной техники и администрирование систем

наименование профиля (специализации)

Обозначение курсовой работы 09.03.04.540000.000 КР Группа ВПР34

Руководитель проекта: профессор Гранков М.В.\_\_\_\_\_\_

подпись (должность, И.О.Ф.)

|  |  |  |
| --- | --- | --- |
| Проект (работа) защищен(а) |  |  |
| дата | оценка | подпись |

Ростов-на-Дону

2022



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет «Информатика и вычислительная техника»

Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

**ЗАДАНИЕ**

к курсовой работе по дисциплине (модулю) базы данных

Обучающийся Лапардин Максим Леонидович

Обозначение проекта КР.540000.000 Группа ВПР34

Срок представления проекта (работы) к защите « » 2022 г.

Содержание индивидуального задания:

Разработать для каталога благотворительных организаций систему базы данных и комплекс программ по управлению.

Руководитель проекта М.В. Гранков

подпись, дата И.О.Ф

Задание принял к исполнению М.Л.Лапардин

подпись, дата И.О.Ф

**Содержание**

**Постановка задачи** 4

**1 Внешняя модель** 6

1.1 Описание представлений пользователей и выделение основных сущностей 6

1.2 Описание функциональных зависимостей 9

1.3 Описание форм обработки и представление сведений о хранимой информации 9

1.4 Дополнительные требования 12

1.5 Модель предметной области в виде схемы "Объекты связи" 12

**2 Логическая модель** 15

2.1 Схемы базовых отношений 15

2.2 Домены атрибутов отношений 16

2.3 Множество функциональных зависимостей 19

2.4 Неприводимое множество функциональных зависимостей 20

2.5 Множество супер-ключей 23

2.6 Множество потенциальных ключей 25

2.7 Выбор первичных ключей 28

2.8 Нормализация отношений предметной области 30

2.9 Предикаты для проверки целостности базы данных 32

2.10 Виртуальные отношения 34

2.11 Реляционные выражения 35

3 Физическая модель 37

**Постановка задачи**

Разработать систему базы данных для приложения донорства крови.

Для достижения поставленной цели необходимо решить следующие задачи:

* составить внешние представления конечных пользователей, описание объектов предметной области, их атрибутов и связей между объектами, функциональных зависимостей, способов, форм обработки и представления сведений о хранимой в БД информации;
* составить логическую модель предметной области с использованием реляционной модели, разработать схемы базовых отношений, описать домены атрибутов всех отношений, построить множество функциональных зависимостей, построить неприводимое множество функциональных зависимостей, построить множество супер – ключей, по множеству супер - ключей построить множество потенциальных ключей, доказать их потенциальность, выбрать первичные ключи, обосновать выбор, провести нормализацию отношений до уровня не ниже 3НФ (на каждом этапе нормализации обосновывать выбор первичных ключей для каждого отношения), разработать предикат для проверки целостности базы данных, описать требуемые виртуальные отношения, разработать реляционные выражения для реализации запросов;
* составить физическую модель, которая должна содержать тексты программ для создания и поддержки всех используемых отношений, визуализацию запросов в виде экранных форм и документов (при реализации внутренней модели особое внимание уделить выполнению следующих правил Кодда: физическая независимость данных; логическая независимость данных; дистрибутивная независимость), разрабатываемое программное обеспечение должно реализовывать сетевой, многопользовательский способ доступа, содержать инструкции по инсталляции программного обеспечения и базовых отношений, требуется обеспечить простое изменение основных констант, характеризующих предметную область;
* разработать сопроводительную документацию.

# **1 Внешняя модель**

Внешняя модель или уровень представлений. На этом уровне предметная область (т.е. та область деятельности, в которой осуществляется разработка данной системы) описывается будущими пользователями БД. Каждый пользователь описывает свое представление о предметной области. При этом, описывается: какие объекты важны в работе пользователя, какими свойствами они обладают, связи между объектами, прочие правила взаимодействия объектов.

Внешний уровень обычно поддерживается приложениями, написанными возможно на различных языках программирования. Этот уровень связан со способами представления данных для различных пользователей.

В данном разделе рассматриваются внешние представления пользователей для базы данных «Библиотека», важные аспекты предметной области, функциональные зависимости.

### **1.1 Описание представлений пользователей и выделение основных сущностей**

В качестве предметной области выбирается информационная система библиотеки, для которой необходима база данных сотрудников и посетителей. С точки зрения проектирования базы данных в данной предметной области имеются следующие конечные пользователи: библиотекарь. Опишем предметную область с точки зрения конечных пользователей.

**Администратор:**

Я должен знать и иметь доступ ко всем данных о всех сотрудниках библиотеки, а также иметь доступ к информации о книгах библиотеки.

В процессе работы я выполняю действия:

* Прочесть данные о библиотекарях
* Изменить учетные данные библиотекарей

**Пользователь:**

Мне необходима информация о всех студентах и учителях, которые оформляли карточку на получение книги.

В процессе работы я выполняю действия:

* Прочесть данные о студентах (группа, факультет), учителях (кафедра), карточках, книгах (автор, издательство, тема, категория), библиотеке
* Создать карточку для студента или учителя
* Добавить книгу в карточку
* Удалить книгу из карточки
* Удалить карточку

Из данных описаний предметной области выделить базовые объекты: Библиотека, Библиотекарь, Группа, Факультет, Студент, Карточка студента, Учитель, Карточка учителя, Кафедра учителя, Автор книги, Категория книги, Издатель книги, Тема книги, Книга.

Описание связей между объектами:

1. Каждый учитель числится только в одной кафедре, а одной в кафедре может работать много учителей. Отсюда возникает связь: Кафедра – Учитель(1:n);
2. Каждая группа имеет только один факультет, а в одном факультете может быть много групп. Отсюда возникает связь: Факультет – Группа(1:n);
3. Каждый студент числится только в одной группе, а в одной группе может быть много студентов. Отсюда возникает связь: Группа – Студент(1:n);
4. Каждый студент может иметь только одну карточку студента, а в одной карточке студента может быть записан только один студент. Отсюда возникает связь: Студент – Карточка студента(1:1);
5. Каждый учитель может иметь только одну карточку учителя, а в одной карточке учителя может быть записан только один учитель. Отсюда возникает связь: Учитель – Карточка учителя(1:1);
6. В каждой карточке студента может быть записан только один библиотекарь, а у каждого библиотекаря может храниться несколько карточек студента. Возникает связь: Библиотекарь – Карточка студента (1:n);
7. В каждой карточке учителя может быть записан только один библиотекарь, а у каждого библиотекаря может храниться несколько карточек учителя. Возникает связь: Библиотекарь – Карточка учителя (1:n);
8. В каждой карточке студента может быть записана только одна библиотека, а в каждой библиотеке может храниться несколько карточек студента. Возникает связь: Библиотека – Карточка студента (1:n);
9. В каждой карточке учителя может быть записана только одна библиотека, а в каждой библиотеке может храниться несколько карточек учителя. Возникает связь: Библиотека – Карточка учителя (1:n);
10. В каждой карточке студента может быть записано несколько книг, а в каждой книге может храниться только одна карточка студента. Возникает связь: Карточка студента – Книга (1:n);
11. В каждой карточке учителя может быть записано несколько книг, а в каждой книге может храниться только одна карточка учителя. Возникает связь: Карточка учителя – Книга (1:n);
12. У каждого автора может быть несколько книг, а у каждой книги только один автор. Возникает связь: Автор – Книга (1:n);
13. У каждого издателя может быть много книг, а у каждой книги только один издатель. Возникает связь: Издательство – Книга (1:n);
14. У каждой категории может быть много книг, а у каждой книги только одна категория. Возникает связь: Категория – Книга (1:n);
15. У каждой темы может быть несколько книг, а у каждой книги только одна тема. Возникает связь: Тема – Книга (1:n);
16. У каждой библиотеки может быть несколько книг, а у каждой книги только одна библиотека. Возникает связь: Библиотека – Книга (1:n);

**1.2 Описание функциональных зависимостей**

Рассмотрим список утверждений, полученных на этапе построения внешней модели:

1. У каждого студента лишь одна группа, а группа относится лишь к одному факультету.
2. В одной карточке студента может быть записан лишь один студент, а в одной карточке учителя - лишь один учитель.
3. В библиотеке может находится больше одного библиотекаря.
4. Каждый учитель может относиться только к одной кафедре.
5. У всех пользователей и администраторов различается логин и пароль аккаунта.
6. В библиотеке может храниться много книг.
7. Количество пользователей должно соответствовать количеству библиотекарей.

### **1.3 Описание форм обработки и представление сведений о хранимой информации**

Запрос на получение всех карточек студентов:

У библиотекаря должна быть возможность просмотра списка карточек студентов, которые зарегистрированы в системе. В информации о списке карточек студентов должны быть собраны самые важные элементы:

1. Дата выдачи карточки
2. Дата окончания срока действия карточки
3. Номер студента
4. Номер книги
5. Номер библиотеки
6. Номер библиотекаря

Представление получения списка всех карточек студентов изображено на рисунке 1.1.

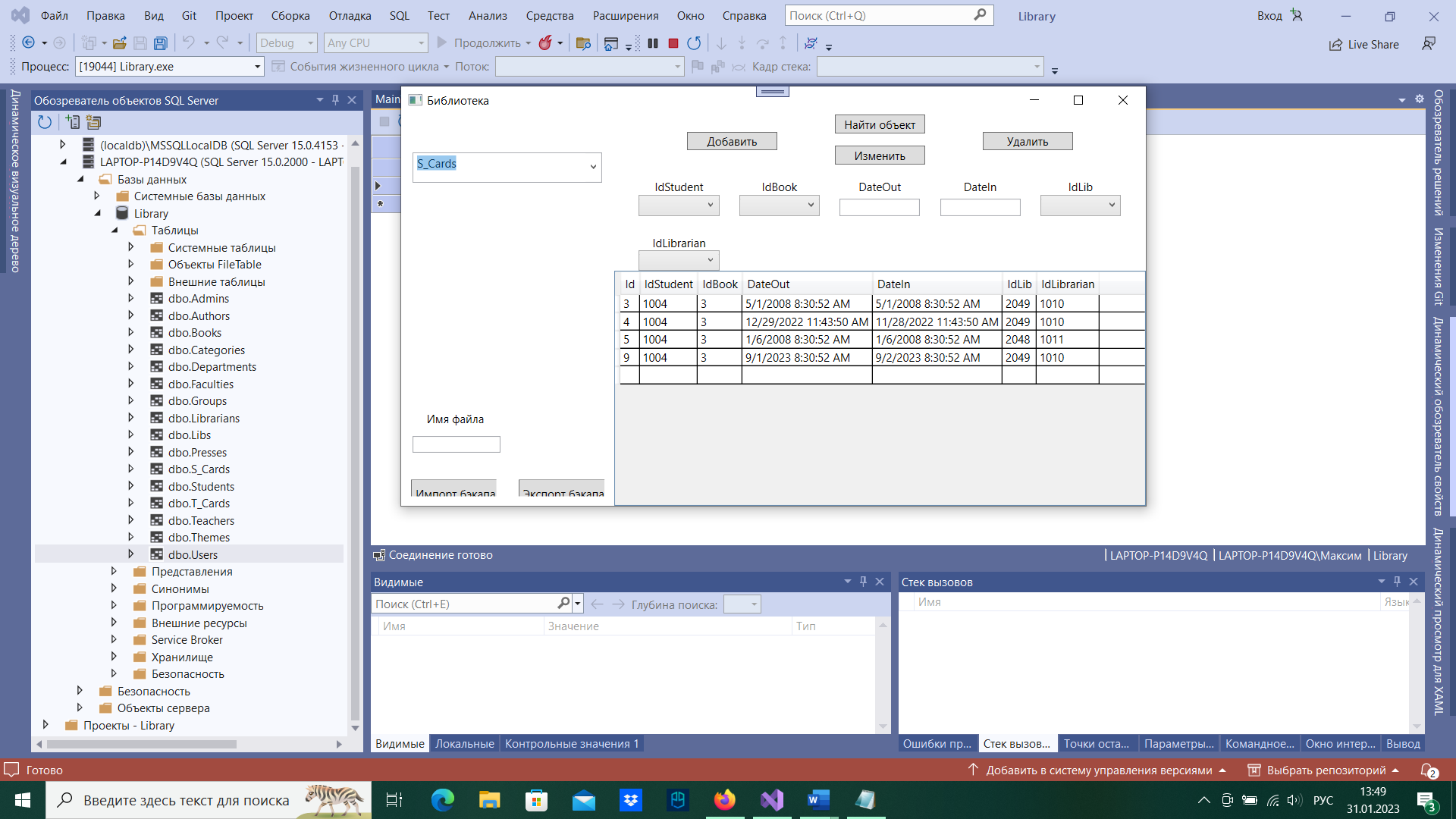


Рисунок 1.1 – представление карточек студентов

У библиотекаря должна быть возможность просмотра списка студентов, которые зарегистрированы в системе. В информации о списке студентов должны быть собраны самые важные элементы:

1. Имя
2. Фамилия
3. Номер группы
4. Номер семестра

Представление получения списка всех карточек студентов изображено на рисунке 1.2.

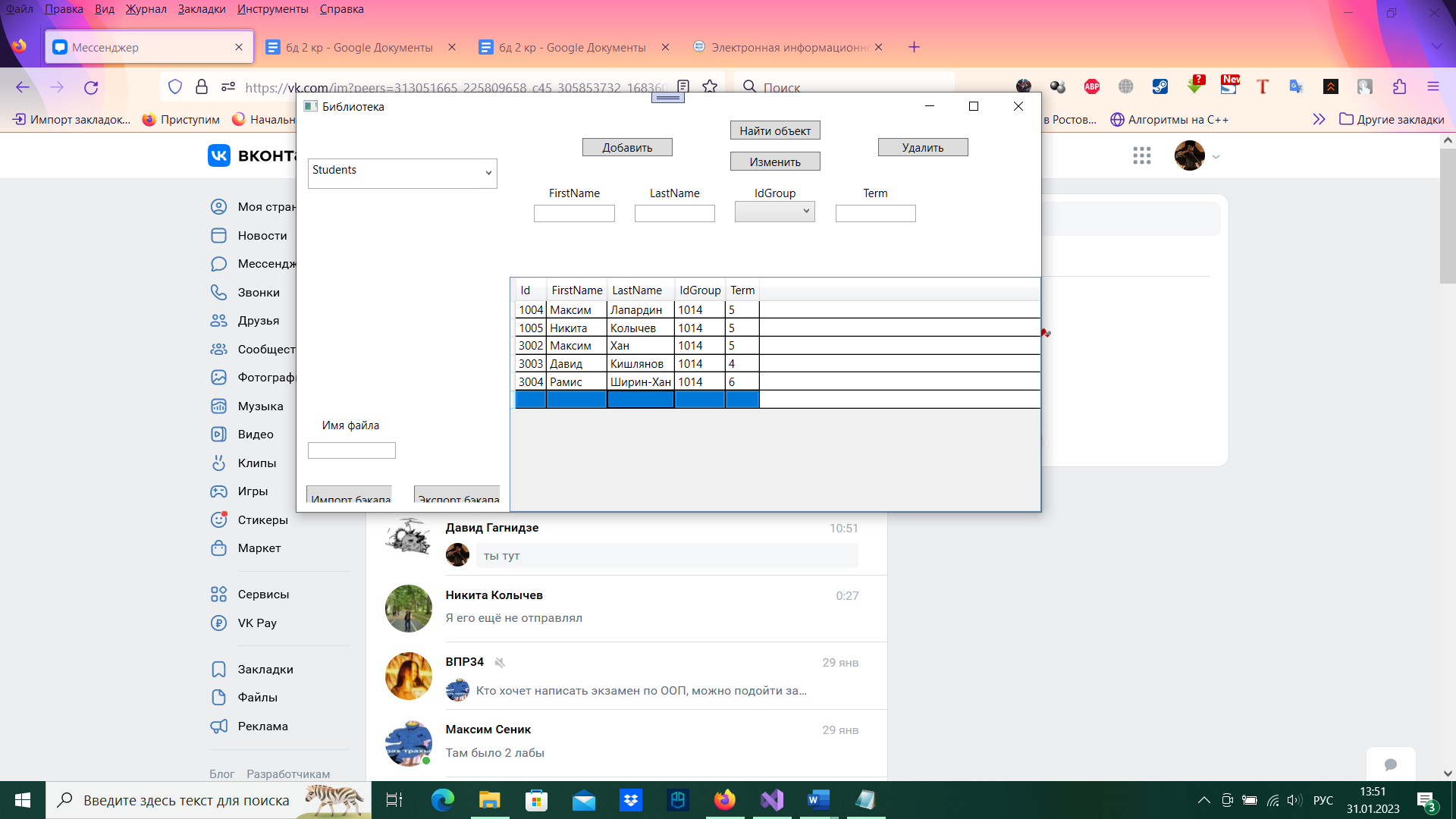


Рисунок 1.2 – представление студентов

У библиотекаря должна быть возможность просмотра списка книг, которые находятся в библиотеке. В информации о списке книг должны быть собраны самые важные элементы:

1. Название
2. Кол-во страниц
3. Год издания
4. Номер темы
5. Номер категории
6. Номер автора
7. Номер издательства
8. Номер библиотеки
9. Комментарий
10. Количество экземпляров

Представление получения списка всех книг изображено на рисунке 1.3.

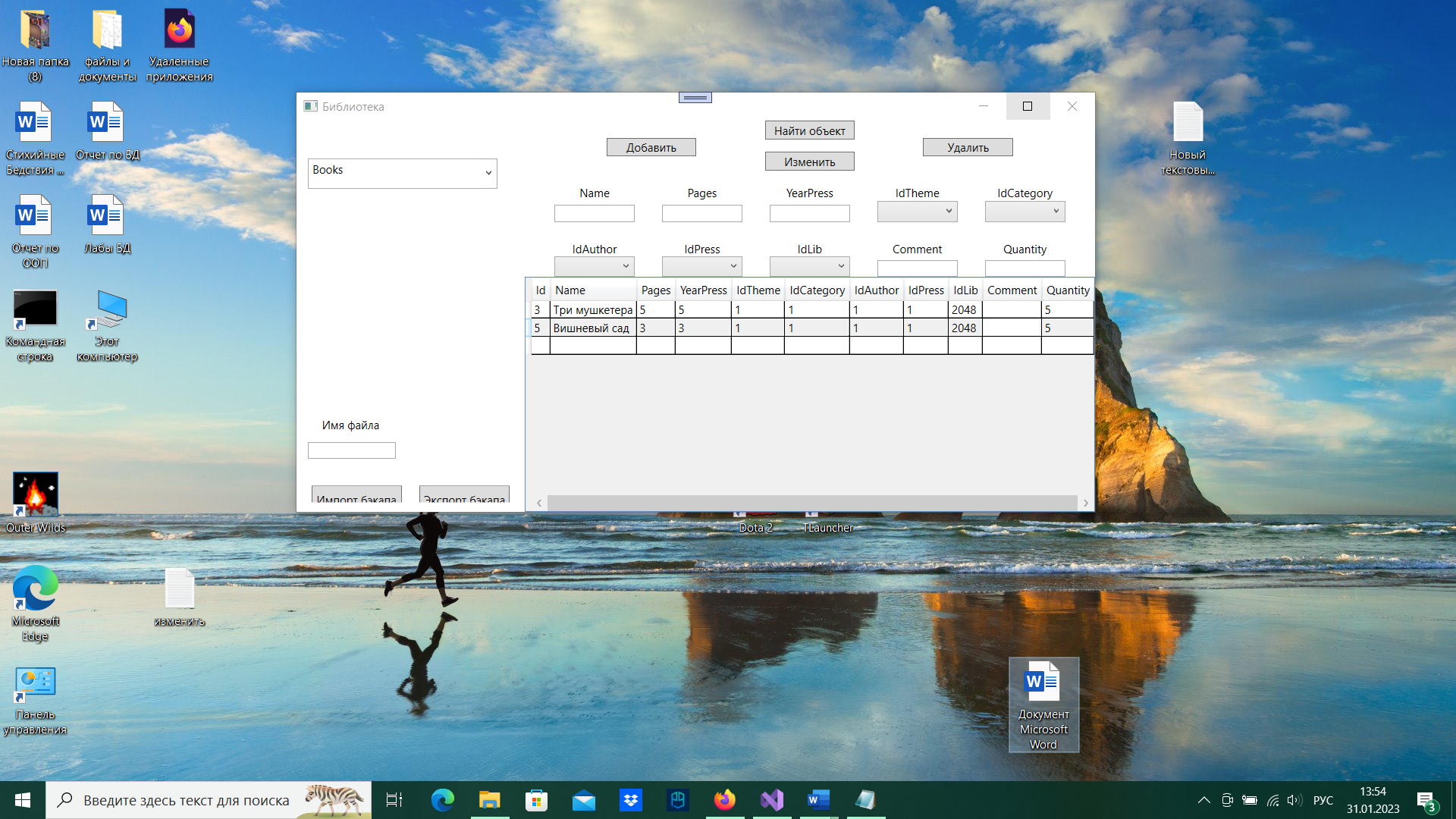


Рисунок 1.3 – представление книг

### **1.4 Дополнительные требования**

### **1.5 Модель предметной области в виде схемы "Объекты связи"**

Важным этапом проектирования базы данных является построение логической схемы. Логическое проектирование прежде всего связано с попыткой представления предметной области в модели БД. В данный момент самой удобной моделью представления является модель «сущность – атрибут» или, как ее еще называют, ER-диаграмма.

Ниже приведены основные понятия ER-диаграмм на примере базы данных «Библиотека».

Сущность – класс однотипных объектов, имеющих одинаковые свойства, информацию о которых необходимо хранить в базе данных. Сущности должны быть различимы. Например, сущностью БД сотрудники является сам сотрудник; не может быть двух абсолютно идентичных сотрудника.

Экземпляр сущности – конкретный представитель данной сущности, то есть сотрудник с определенной датой приема, заработной платой и стажем.

Атрибут сущности – некая поименованная характеристика сущности.

Связь – ассоциирование нескольких сущностей. Сущность также может ассоциироваться сама с собой. Наличие связей между сущностями позволяет пользователям отыскать одну запись через другую.

Связи могут быть следующих типов:

- один-к-одному, то есть один экземпляр первой сущности связан с одним экземпляром второй сущности;

- один-ко-многим, что один экземпляр первой сущности связан с несколькими экземплярами второй сущности.

- многие-ко-многим означает, что каждый экземпляр первой сущности может быть связан с несколькими экземплярами второй сущности, и каждый экземпляр второй сущности может быть связан с несколькими экземплярами первой сущности.

Итак, на основе представлений конечных пользователей БД «Библиотека» можно составить обобщенное представление предметной области, выделив основные объекты будущей базы данных и отбросив второстепенные. В качестве концептуальной модели была выбрана ER- диаграмма. ER-диаграммы удобны тем, что процесс выделения сущностей, атрибутов и связей является цикличным. Разработав первый приближенный вариант диаграмм, разработчик базы данных уточняет их, опрашивая экспертов предметной области – будущих конечных пользователей. ER-модель использует графическое изображение сущностей предметной области, их свойств (атрибутов), и взаимосвязей между сущностями.

Главными сущностями БД «Библиотека» будут следующие: «Библиотека», «Библиотекарь», «Группа», «Факультет», «Студент», «Карточка студента», «Учитель», «Карточка учителя», «Кафедра учителя», «Автор книги», «Категория книги», «Издатель книги», «Тема книги», «Книга». Рассмотрим атрибуты каждой сущности, необходимые конечным пользователям для работы:

«Библиотека»: название.

«Библиотекарь»: имя, фамилия.

«Группа»: название.

«Факультет»: название.

«Студент»: имя, фамилия, семестр.

«Карточка студента»: дата выдачи, дата окончания срока действия.

«Учитель»: имя, фамилия.

«Карточка учителя»: дата выдачи, дата окончания срока действия.

«Кафедра учителя»: название.

«Автор книги»: имя, фамилия.

«Категория книги»: название.

«Издатель книги»: название.

«Тема книги»: название.

«Книга»: имя, кол-во страниц, год издание, комментарий, количество экземпляров данной книги.

Основные связи в проектируемой ER-диаграмме:

«Библиотека» числится в «Карточке студента» и «Карточке учителя»; «Библиотекарь» числится в «Библиотеке», «Карточке студента» и «Карточке учителя»; «Группа» имеет «Факультет»; «Студент» числится в «Группе» и «Карточке студента»; «Учитель» числится на «Кафедре» и в «Карточке учителя»; «Книга имеет «Автора», «Категорию», «Издателя», «Тему»; «Книга» числится в «Библиотеке», «Карточке студента» и «Карточке Учителя».

Таким образом, на основе приведенных представлений пользователей, выделенных основных объектов и описанных атрибутов и связей была спроектирована семантическая модель предметной области «Библиотека» (Приложение А – ER-диаграмма).

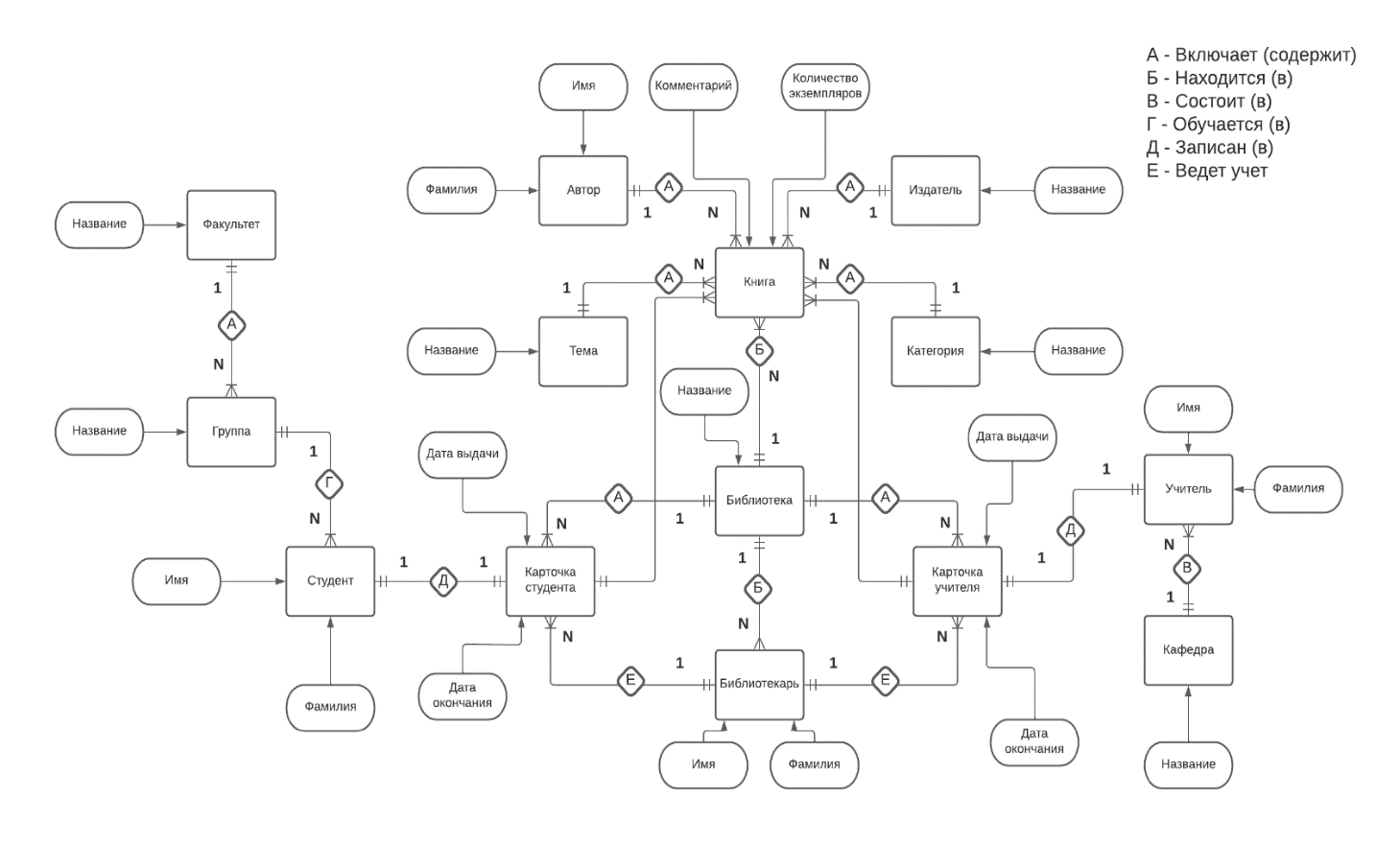


Рисунок 1.4 – модель предметной области информационной системы библиотеки в виде схемы “Объекты связи

# **2 Логическая модель**

Концептуальный или логический уровень. На этом уровне прикладными программистами разрабатывается обобщенное описание предметной области, которое опирается на представления пользователей. В логической модели используются один из формальных языков. Выбор языка определяется моделью используемой в СУБД.

Концептуальный уровень является обобщенным представлением всех пользователей. Может быть несколько внешних представлений, каждое из которых состоит из представления определенной части базы данных, но может быть только одно концептуальное представление, состоящее из абстрактного представления базы данных в целом.

### **2.1 Схема базовых отношений**

Карточка студента::={**номер карточки студента**, дата выдачи, дата окончания, студент::={**номер студента**, имя, фамилия, группа::={**номер группы**, название, факультет::={**номер факультета**, название}}, номер семестра}, книга::= {**номер книги**, название, кол-во страниц, год издания, автор::={**номер автора**, имя, фамилия}, тема::={**номер темы**, название}, категория::={**номер категории**, название}, издательство::={**номер издательства**, название}, библиотека::={**номер библиотеки**, название}, комментарий, кол-во экземпляров}, библиотека::={**номер библиотеки**, название}, библиотекарь::={номер библиотекаря, имя, фамилия, библиотека::={**номер библиотеки**, название}}}

Карточка учителя::={**номер карточки учителя**, дата выдачи, дата окончания, учитель::={**номер учителя**, имя, фамилия, кафедра::={**номер кафедры**, название}}, книга::= {**номер книги**, название, кол-во страниц, год издания, автор::={**номер автора**, имя, фамилия}, тема::={**номер темы**, название}, категория::={**номер категории**, название}, издательство::={**номер издательства**, название}, библиотека::={**номер библиотеки**, название}, комментарий, кол-во экземпляров}, библиотека::={**номер библиотеки**, название}, библиотекарь::={номер библиотекаря, имя, фамилия, библиотека::={**номер библиотеки**, название}}}

### **2.2 Домены атрибутов отношений**

**Отношение «Библиотека»:**

* Номер - целое неотрицательное число
* Название - строка до 50 символов

**Отношение «Библиотекарь»:**

* Номер - целое неотрицательное число
* Имя - строка до 25 символов
* Фамилия - строка до 35 символов
* Номер библиотеки - целое неотрицательное число

**Отношение «Группа»:**

* Номер - целое неотрицательное число
* Название - строка до 40 символов
* Номер факультета - целое неотрицательное число

**Отношение «Факультет»:**

* Номер - целое неотрицательное число
* Название - строка до 40 символов

**Отношение «Студент»:**

* Номер - целое неотрицательное число
* Имя - строка до 25 символов
* Фамилия - строка до 35 символов
* Номер Группы - целое неотрицательное число
* Номер семестра - целое неотрицательное число

**Отношение «Карточка студента»:**

* Номер - целое неотрицательное число
* Номер студента - целое неотрицательное число
* Номер книги - целое неотрицательное число
* Дата выдачи - дата и время
* Дата окончания - дата и время
* Номер библиотеки - целое неотрицательное число
* Номер библиотекаря - целое неотрицательное число

**Отношение «Учитель»:**

* Номер - целое неотрицательное число
* Имя - строка до 25 символов
* Фамилия - строка до 35 символов
* Номер Кафедры - целое неотрицательное число

**Отношение «Карточка учителя»:**

* Номер - целое неотрицательное число
* Номер учителя - целое неотрицательное число
* Номер книги - целое неотрицательное число
* Дата выдачи - дата и время
* Дата окончания - дата и время
* Номер библиотеки - целое неотрицательное число
* Номер библиотекаря - целое неотрицательное число

**Отношение «Кафедра»:**

* Номер - целое неотрицательное число
* Название - строка до 40 символов

**Отношение «Автор»:**

* Номер - целое неотрицательное число
* Имя - строка до 25 символов
* Фамилия - строка до 35 символов

**Отношение «Категория»:**

* Номер - целое неотрицательное число
* Название - строка до 30 символов

**Отношение «Издатель»:**

* Номер - целое неотрицательное число
* Название - строка до 30 символов

**Отношение «Тема»:**

* Номер - целое неотрицательное число
* Название - строка до 30 символов

**Отношение «Книга»:**

* Номер - целое неотрицательное число
* Название - строка до 40 символов
* Кол-во страниц - целое неотрицательное число
* Год издания - целое неотрицательное число
* Номер темы - целое неотрицательное число
* Номер категории - целое неотрицательное число
* Номер автора - целое неотрицательное число
* Номер издательства - целое неотрицательное число
* Номер библиотеки - целое неотрицательное число
* Комментарий - строка до 50 символов
* Количество экземпляров - целое неотрицательное число

### **2.3 Множество функциональных зависимостей.**

На основании списка утверждений, полученного после этапа построения внешней модели, описанного в главе 1.2, построим множество функциональных зависимостей:

S={

{Номер библиотеки} → {Название},

{Номер библиотекаря} → {Имя, Фамилия, Номер библиотеки},

{Номер группы} → {Название, Номер факультета},

{Номер факультета} → {Название},

{Номер студента} → {Имя, Фамилия, Номер группы, Номер семестра},

{Номер карточки студента} → {Номер студента, Номер книги, Номер библиотеки, Номер библиотекаря, Дата выдачи, Дата окончания},

{Номер учителя} → {Имя, Фамилия, Номер кафедры},

{Номер карточки учителя} → {Номер учителя, Номер книги, Номер библиотеки, Номер библиотекаря, Дата выдачи, Дата окончания},

{Номер кафедры} → {Название},

{Номер автора} → {Имя, Фамилия},

{Номер категории} → {Название},

{Номер издательства} → {Название},

{Номер темы} → {Название},

{Номер книги} → {Название, Кол-во страниц, Год издания, Кол-во экземпляров, Номер темы, Номер категории, Номер автора, Номер издательства, Номер библиотеки, Комментарий}

}

### **2.4 Неприводимое множество функциональных зависимостей.**

На основе множества функциональных зависимостей S, описанного в разделе 2.3, путем равносильных преобразований по правилу Амстронга, преобразуем множество ФЗ в неприводимое множество ФЗ.

S0 = {

{нб} → {назб},

{нбиб} → {имя, фам, нб},

{нг} → {назг, нф},

{нф} → {назф},

{нс} → {имя, фам, нг, нсем},

{нкс} → {нс, нк, дв, до, нб, нбиб},

{дв, до, нс} → {нкс, нб, нк, нбиб}

{нуч} → {нмя, фам, нкаф},

{нкуч} → {нуч, нк, дв, до, нб, нбиб},

{дв, до, нуч} → {нкуч, нб, нк, нбиб}

{нкаф} → {назкаф},

{навт} → {назавт},

{нкат} → {назкат},

{низд} → {назизд},

{нт} → {назт},

{нк} → {назк, кс, гизд, кэ, нт, нкат, навт, низд, нб, ком}

}

Раскрываем определяемые множества с помощью правила декомпозиции.

S1 = {

{нб} → {назб},

{нбиб} → {имя}, {нбиб} → {фам}, {нбиб} → {нб},

{нг} → {назг}, {нг} → {нф},

{нф} → {назф},

{нс} → {имя}, {нс} → {фам}, {нс} → {нг}, {нс} → {нсем},

{нкс} → {нс}, {нкс} → {нк}, {нкс} → {дв}, {нкс} → {до}, {нкс} → {нб}, {нкс} → {нбиб},

{дв, до, нс} → {нкс}, {дв, до, нс} → {нб}, {дв, до, нс} → {нк}, {дв, до, нс} → {нбиб},

{нуч} → {нмя}, {нуч} → {фам}, {нуч} → {нкаф},

{нкуч} → {нуч}, {нкуч} → {нк}, {нкуч} → {дв}, {нкуч} → {до}, {нкуч} → {нб}, {нкуч} → {нбиб},

{дв, до, нуч} → {нкуч}, {дв, до, нуч} → {нб}, {дв, до, нуч} → {нк}, {дв, до, нуч} → {нбиб},

{нкаф} → {назкаф},

{навт} → {назавт},

{нкат} → {назкат},

{низд} → {назизд},

{нт} → {назт},

{нк} → {назк}, {нк} → {кс}, {нк} → {гизд}, {нк} → {кэ}, {нк} → {нт}, {нк} → {нкат}, {нк} → {навт}, {нк} → {низд}, {нк} → {нб}, {нк} → {ком}

}

По правилу транзитивности объединим:

{нс} → {нг} и {нг} → {нф} = {нс} → {нф}

{нк} → {нб} и {нб} → {назб} = {нк} → {назб}

{нкс} → {нбиб} и {нбиб} → {имя} = {нкс} → {имя}

{нк} → {навт} и {навт} → {назавт} = {нк} → {назавт}

{нк} → {нкат} и {нкат} → {назкат} ={нк} → {назкат}

{нк} → {нт} и {нт} → {назт} = {нк} → {назт}

{нк} → {низд} и {низд} → {назизд} = {нк} → {назизд}

{нуч} → {нкаф} и {нкаф} → {назкаф} = {нуч} → {назкаф}

{нкуч} → {нуч} и {нуч} → {нмя} = {нкуч} → {нмя}

{нкуч} → {нбиб} и {нбиб} → {нб} = {нкуч} → {нб}

{нкс} → {нс} и {нс} → {нсем} = {нкс} → {нсем}

{нкс} → {нк} и {нк} → {назк} = {нкс} → {назк}

{нкуч} → {нк} и {нк} → {кс} = {нкуч} → {кс}

{дв, до, нс} → {нкс} и {нкс} → {дв} = {дв, до, нс} → {дв}

{дв, до, нс} → {нк} и {нк} → {кэ} = {дв, до, нс} → {кэ}

{дв, до, нс} → {нбиб} и{нбиб} → {фам} = {дв, до, нс} → {фам}

{дв, до, нуч} → {нкуч} и {нкуч} → {дв} = {дв, до, нуч} → {дв}

{дв, до, нуч} → {нк} и {нк} → {ком} = {дв, до, нуч} → {ком}

S2 = {

{нс} → {нф}, {нс} → {имя}, {нс} → {фам},

{нк} → {назб}, {нк} → {назавт}, {нк} → {назкат}, {нк} → {назт}, {нк} → {назизд}, {нк} → {гизд},

{нуч} → {назкаф}, {нуч} → {фам},

{нкуч} → {нмя}, {нкуч} → {нб}, {нкуч} → {кс}, {нкуч} → {до}, {нкуч} → {нб},

{нкс} → {нсем}, {нкс} → {назк}, {нкс} → {имя}, {нкс} → {до}, {нкс} → {нб},

{дв, до, нс} → {дв}, {дв, до, нс} → {нб}, {дв, до, нс} → {кэ}, {дв, до, нс} → {фам},

{дв, до, нуч} → {дв}, {дв, до, нуч} → {нб}, {дв, до, нуч} → {ком}, {дв, до, нуч} → {нбиб},

{нг} → {назг},

{нф} → {назф},

}

По правилу транзитивности объединим:

{нс} → {нф} и {нф} → {назф} = {нс} → {назф}

S3 = {

{нс} → {назф}, {нс} → {имя}, {нс} → {фам},

{нг} → {назг},

{нк} → {назб}, {нк} → {назавт}, {нк} → {назкат}, {нк} → {назт}, {нк} → {назизд}, {нк} → {гизд},

{нуч} → {назкаф}, {нуч} → {фам},

{нкуч} → {нмя}, {нкуч} → {нб}, {нкуч} → {кс}, {нкуч} → {до}, {нкуч} → {нб},

{нкс} → {нсем}, {нкс} → {назк}, {нкс} → {имя}, {нкс} → {до}, {нкс} → {нб},

{дв, до, нс} → {дв}, {дв, до, нс} → {нб}, {дв, до, нс} → {кэ}, {дв, до, нс} → {фам},

{дв, до, нуч} → {дв}, {дв, до, нуч} → {нб}, {дв, до, нуч} → {ком}, {дв, до, нуч} → {нбиб},

}

Дальнейшее преобразование множества функциональных зависимостей без изменения замыкания множества функциональной зависимости невозможно, S3 является неприводимым множеством.

**2.5 Множество суперключей**

Суперключом отношения R называется множество атрибутов отношения R, которое содержит как подмножество хотя бы один потенциальный ключ (т.е. ключ, который обладает свойством уникальности). Не существует двух [кортежей](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D1%80%D1%82%D0%B5%D0%B6_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) данного отношения, в которых значения этого множества атрибутов совпадают (равны).

Алгоритм получения суперключа:

Пусть известно некоторое множество S функциональных зависимостей отношения R. Требуется найти потенциальные ключи этого отношения. Если построить множество суперключей (каждый из элементов этого множества содержит потенциальный ключ) и исключить из множества суперключей ключи, обладающие свойством избыточности, то мы получим множество потенциальных ключей. Чтобы установить является ли подмножество атрибутов Кi супеключом необходимо выяснить являются ли функционально зависимыми все атрибуты отношения R от Ki, т. е. если для одного отношения R, с множеством атрибутов А и множеством ФЗ S и множеством Ki ⊂ A удается найти множество К +i ⊆ А такие, что все элементы К +i функционально зависят от Кi и при этом К +i = А, то Кi является суперключом (и т.о., возможно, является потенциальным ключом).

Определим суперключ для отношения Карточка студента.

Имеется множество атрибутов B = {нкс, нс, нк, дв, до, нб, нбиб}, подмножество ФЗ {{нкс} → {нс}, {нкс} → {нк}, {нкс} → {дв}, {нкс} → {до}, {нкс} → {нб}, {нкс} → {нбиб}, {дв, до, нс} → {нб} , {дв, до, нс} → {нкс}, {дв, до, нс} → {нк}, {дв, до, нс} → {нбиб}}

Определим является ли множество {дв, до, нс} cуперключом:  
Построим замыкание X+ для множества {дв, до, нс}:

X+ = {дв, до, нс}

Переберем все ФЗ подмножества:

1. {дв, до, нс} → {нб} ^ {дв, до, нс} X+ {нб} X+ → X+ = {дв, до, нс, нб}
2. {нкс} → {нб} {нб} X+
3. {дв, до, нс} → {нк} ^ {дв, до, нс} X+ {нк} X+ → X+ = {дв, до, нс, нб, нк}
4. {дв, до, нс} → {нкс} ^ {дв, до, нс} X+ {нкс} X+ → X+ = {дв, до, нс, нб, нк, нкс}
5. {дв, до, нс} → {нбиб} ^ {дв, до, нс} X+ {нбиб} X+ → X+ = {дв, до, нс, нб, нк, нкс, нбиб}

Далее X+ не изменится и т.о. X+ = {нкс, нс, нк, дв, до, нб, нбиб}

Множество X+ = B и все атрибуты X+ функционально зависимы от {дв, до, нс} => {дв, до, нс} – является суперключом.

Также определим является ли множество {нкс} суперключом:

X+ = {нкс}

Переберем все ФЗ подмножества:

1. {дв, до, нс} → {нб} ^ {дв, до, нс} X+
2. {нкс} → {нб} {нкс} X+ {нб} X+ → X+ = {нкс, нб}
3. {дв, до, нс} → {нк} {дв, до, нс} X+
4. {дв, до, нс} → {нкс} {дв, до, нс} X+
5. {нкс} → {нк} {нкс} X+ {нк} X+ → X+ = {нкс, нб, нк}
6. {нкс} → {нс} {нкс} X+ {нс} X+ → X+ = {нкс, нб, нк, нс}
7. {нкс} → {дв} {нкс} X+ {дв} X+ → X+ = {нкс, нб, нк, нс, дв}
8. {нкс} → {до} {нкс} X+ {до} X+ → X+ = {нкс, нб, нк, нс, дв, до}
9. {нкс} → {нбиб} {нкс} X+ {нбиб} X+ → X+ = {нкс, нб, нк, нс, дв, до, нбиб}

Далее X+ не изменится и т.о. X+ = {нкс, нб, нк, нс, дв, до, нбиб}

Множество X+ = B и все атрибуты X+ функционально зависимы от   
{нкс} => {нкс} – является суперключом.

Таким образом для отношения Карточка студента множеством суперключей является: {{нкс}, {дв, до, нс}}.

**2.6 Множество потенциальных ключей**

Пусть R – некоторая переменная отношения, тогда потенциальный ключ K для R это подмножество атрибутов R, всегда обладающее следующими свойствами:

1. Свойство уникальности: нет двух различных кортежей в текущем значении переменной R с одинаковыми значениями K
2. Свойство не избыточности: никакое из подмножеств K не обладает свойством уникальности.

Пусть известно некоторое множество S функциональных зависимостей отношения R. Требуется найти потенциальные ключи этого отношения. Если построить множество суперключей (каждый из элементов этого множества содержит потенциальный ключ) и исключить из множества суперключей ключи, обладающие свойством избыточности, то мы получим множество потенциальных ключей.

Определим множество потенциальных ключей отношения Карточка студента:

Потенциальным ключом является суперключ {дв, до, нс}, который был определен в пункте 2.5.

Проверим суперключ {дв, до, нс} на неизбыточность. Для этого проверим все подмножества данного ключа на уникальность:

1. Построим замыкание X+ для множества {дв, до}:

X+={дв, до}

ФЗ для данного подмножества отсутствуют (значения атрибутов одного кортежа могут совпадать со значениями атрибутов другого кортежа), следовательно, и подмножество {дв, до} не обладает свойством уникальности.

1. Построим замыкание X+ для множества {дв, нс}

X+={дв, нс}

ФЗ для данного подмножества отсутствуют (значения атрибутов одного кортежа могут совпадать со значениями атрибутов другого кортежа), следовательно, и подмножество {дв, нс} не обладает свойством уникальности.

1. Построим замыкание X+ для множества {до, нс}

X+={до, нс}

ФЗ для данного подмножества отсутствуют (значения атрибутов одного кортежа могут совпадать со значениями атрибутов другого кортежа), следовательно, и подмножество {до, нс} не обладает свойством уникальности.

1. Построим замыкание X+ для множества {дв}

X+={дв}

ФЗ для данного подмножества отсутствуют (значения атрибутов одного кортежа могут совпадать со значениями атрибутов другого кортежа), следовательно, и подмножество {дв} не обладает свойством уникальности.

1. Построим замыкание X+ для множества {до}

X+={до}

ФЗ для данного подмножества отсутствуют (значения атрибутов одного кортежа могут совпадать со значениями атрибутов другого кортежа), следовательно, и подмножество {до} не обладает свойством уникальности.

1. Построим замыкание X+ для множества {нс}

Имеется множество атрибутов С = {нкс, нс, нк, дв, до, нб, нбиб}, подмножество ФЗ { {дв, до, нс} → {нб} , {дв, до, нс} → {нкс}, {дв, до, нс} → {нк}, {дв, до, нс} → {нбиб}, {нс} → {имя}, {нс} → {фам}, {нс} → {нг}, {нс} → {нсем}}

X+={нс}

Переберем все ФЗ подмножества:

1. {нс} → {имя} ^ {нс} X+ {имя} X+ → X+ = {нс, имя}
2. {нс} → {фам} ^ {нс} X+ {фам} X+ → X+ = {нс, имя, фам}
3. {нс} → {нг} ^ {нс} X+ {нг} X+ → X+ = {нс, имя, фам, нг}
4. {нс} → {нсем} ^ {нс} X+ {нсем} X+ → X+ = {нс, имя, фам, нг, нсем}

Далее X+ не изменится и т.о. X+ = {нс, имя, фам, нг, нсем}

Множество X+ ≠ С и все атрибуты X+ функционально зависимы от   
{нс} => {нс} – не является уникальным.

Суперключ {дв, до, нс} обладает свойством неизбыточности, т.к. ни одно подмножество данного суперключа не обладает свойством уникальности.

**2.7 Выбор первичных ключей**

Ключ (первичный ключ) – множество атрибутов, значение которых уникальным образом идентифицирует кортеж в отношении.

Первичным ключом отношения **библиотека** выбирается атрибут **номер библиотеки**, потому что данный атрибут уникален для каждого кортежа отношения, определяет все множество атрибутов отношения, то есть является потенциальным ключом, имеющий наименьшую мощность множества.

Первичным ключом отношения **библиотекарь** выбирается атрибут **номер библиотекаря**, потому что данный атрибут уникален для каждого кортежа отношения, определяет все множество атрибутов отношения, то есть является потенциальным ключом, имеющий наименьшую мощность множества.

Первичным ключом отношения **группа** выбирается атрибут **номер группы**, потому что данный атрибут уникален для каждого кортежа отношения, определяет все множество атрибутов отношения, то есть является потенциальным ключом, имеющий наименьшую мощность множества.

Первичным ключом отношения **факультет** выбирается атрибут **номер факультета**, потому что данный атрибут уникален для каждого кортежа отношения, определяет все множество атрибутов отношения, то есть является потенциальным ключом, имеющий наименьшую мощность множества.

Первичным ключом отношения **студент** выбирается атрибут **номер студента**, потому что данный атрибут уникален для каждого кортежа отношения, определяет все множество атрибутов отношения, то есть является потенциальным ключом, имеющий наименьшую мощность множества.

Первичным ключом отношения **карточка студента** выбирается множество атрибутов **номер карточки студента**, потому что данное множество атрибутов уникально для каждого кортежа отношения, определяет все множество атрибутов отношения, то есть является потенциальным ключом, имеющим наименьшую мощность множества.

Первичным ключом отношения **учитель** выбирается атрибут **номер учителя**, потому что данный атрибут уникален для каждого кортежа отношения, определяет все множество атрибутов отношения, то есть является потенциальным ключом, имеющий наименьшую мощность множества.

Первичным ключом отношения **карточки учителя** выбирается атрибут **номер карточки учителя**, потому что данный атрибут уникален для каждого кортежа отношения, определяет все множество атрибутов отношения, то есть является потенциальным ключом, имеющий наименьшую мощность множества.

Первичным ключом отношения **кафедра** выбирается атрибут **номер кафедры**, потому что данный атрибут уникален для каждого кортежа отношения, определяет все множество атрибутов отношения, то есть является потенциальным ключом, имеющий наименьшую мощность множества.

Первичным ключом отношения **автор** выбирается атрибут **номер автора**, потому что данный атрибут уникален для каждого кортежа отношения, определяет все множество атрибутов отношения, то есть является потенциальным ключом, имеющий наименьшую мощность множества.

Первичным ключом отношения **категория** выбирается атрибут **номер категории**, потому что данный атрибут уникален для каждого кортежа отношения, определяет все множество атрибутов отношения, то есть является потенциальным ключом, имеющий наименьшую мощность множества.

Первичным ключом отношения **издательство** выбирается атрибут **номер издательства**, потому что данный атрибут уникален для каждого кортежа отношения, определяет все множество атрибутов отношения, то есть является потенциальным ключом, имеющий наименьшую мощность множества.

Первичным ключом отношения **тема** выбирается атрибут **номер темы**, потому что данный атрибут уникален для каждого кортежа отношения, определяет все множество атрибутов отношения, то есть является потенциальным ключом, имеющий наименьшую мощность множества.

Первичным ключом отношения **книга** выбирается атрибут **номер книги**, потому что данный атрибут уникален для каждого кортежа отношения, определяет все множество атрибутов отношения, то есть является потенциальным ключом, имеющий наименьшую мощность множества.

### **2.8 Нормализация отношений предметной области**

**Исходное отношение**

Карточка студента::={**номер карточки студента**, дата выдачи, дата окончания, студент::={**номер студента**, имя, фамилия, группа::={**номер группы**, название, факультет::={**номер факультета**, название}}, номер семестра}, книга::= {**номер книги**, название, кол-во страниц, год издания, автор::={**номер автора**, имя, фамилия}, тема::={**номер темы**, название}, категория::={**номер категории**, название}, издательство::={**номер издательства**, название}, библиотека::={**номер библиотеки**, название}, комментарий, кол-во экземпляров}, библиотека::={**номер библиотеки**, название}, библиотекарь::={номер библиотекаря, имя, фамилия, библиотека::={**номер библиотеки**, название}}}

Карточка учителя::={**номер карточки учителя**, дата выдачи, дата окончания, учитель::={**номер учителя**, имя, фамилия, кафедра::={**номер кафедры**, название}}, книга::= {**номер книги**, название, кол-во страниц, год издания, автор::={**номер автора**, имя, фамилия}, тема::={**номер темы**, название}, категория::={**номер категории**, название}, издательство::={**номер издательства**, название}, библиотека::={**номер библиотеки**, название}, комментарий, кол-во экземпляров}, библиотека::={**номер библиотеки**, название}, библиотекарь::={номер библиотекаря, имя, фамилия, библиотека::={**номер библиотеки**, название}}}

**Приводим отношения к 1НФ**

Раскрываем составные атрибуты.

Отношение находится в первой нормальной форме (1НФ), если каждый его атрибут атомарен, т.е. содержит только одно значение, а не список значений.

Библиотека::={**номер библиотеки**, название}

Библиотекарь::={**номер библиотекаря**, имя, фамилия, номер библиотеки}

Группа::={**номер группы**, название, номер факультета}

Факультет::={**номер факультета**, название}

Студент::={**номер студента**, имя, фамилия, номер группы, номер семестра}

Карточка студента::={**номер карточки студента**, дата выдачи, дата окончания, номер студента, номер книги, номер библиотеки, номер библиотекаря}

Учитель::={**номер учителя**, имя, фамилия, номер кафедры},

Карточка учителя::={**номер карточки учителя**, дата выдачи, дата окончания, номер учителя, номер книги, номер библиотеки, номер библиотекаря}

Кафедра::={**номер кафедры**, название},

Автор::= {**номер автора**, имя, фамилия},

Категория::={**номер категории**, название},

Издательство::= {**номер издательства**, название},

Тема::={**номер темы**, название},

Книга::={**номер книги**, название, кол-во страниц, год издания, кол-во экземпляров, номер темы, номер категории, номер автора, номер издательства, номер библиотеки, комментарий}

**Приводим отношения к 2НФ**

Так как отношение факультет используется только в отношении группа и потенциально меняться будет редко, разумно объединить отношение фкультет и группа через ключ номер факультета в отношении пользователь.

Группа::={**номер группы**, название, факультет}

Так как отношение группа используется только в отношении студент и потенциально меняться будет редко, разумно объединить отношение группа и студент через ключ номер группы в отношении студент.

Студент::={**номер студента**, имя, фамилия, группа, номер семестра}

Так как отношение кафедра используется только в отношении учитель и потенциально меняться будет редко, разумно объединить отношение кафедра и учитель через ключ номер кафедры в отношении учитель.

Учитель::={**номер учителя**, имя, фамилия, кафедра}

**Приводим отношения к 3НФ**

Во всех отношениях выполняется нетранзитивная независимость, то есть все неключевые атрибуты взаимно независимы, и при изменении одного из атрибутов, остальные атрибуты останутся неизменными.

### **2.9 Предикаты для проверки целостности базы данных**

**Условия целостности для отношения Библиотека:**

Номер библиотеки уникальным образом идентифицирует кортеж в отношении, уникален для каждого кортежа.

Все атрибуты должны иметь значение.

**Условия целостности для отношения Библиотекарь:**

Номер библиотекаря уникальным образом идентифицирует кортеж в отношении, уникален для каждого кортежа.

Все атрибуты должны иметь значение.

**Условия целостности для отношения Группа:**

Номер группа уникальным образом идентифицирует кортеж в отношении, уникален для каждого кортежа.

Все атрибуты должны иметь значение.

**Условия целостности для отношения Факультет:**

Номер факультета уникальным образом идентифицирует кортеж в отношении, уникален для каждого кортежа.

Все атрибуты должны иметь значение.

**Условия целостности для отношения Студент:**

Номер студента уникальным образом идентифицирует кортеж в отношении, уникален для каждого кортежа.

Номер семестра не может быть меньше 0.

Все атрибуты должны иметь значение.

**Условия целостности для отношения Карточка студента:**

Номер карточки студента уникальным образом идентифицирует кортеж в отношении, уникален для каждого кортежа.

Дата выдачи должна быть раньше даты окончания.

Все атрибуты должны иметь значение.

**Условия целостности для отношения Учитель:**

Номер учителя уникальным образом идентифицирует кортеж в отношении, уникален для каждого кортежа.

Все атрибуты должны иметь значение.

**Условия целостности для отношения Карточка учителя:**

Номер карточки учителя уникальным образом идентифицирует кортеж в отношении, уникален для каждого кортежа.

Дата выдачи должна быть раньше даты окончания.

Все атрибуты должны иметь значение.

**Условия целостности для отношения Кафедра:**

Номер кафедры уникальным образом идентифицирует кортеж в отношении, уникален для каждого кортежа.

Все атрибуты должны иметь значение.

**Условия целостности для отношения Автор:**

Номер автора уникальным образом идентифицирует кортеж в отношении, уникален для каждого кортежа.

Все атрибуты должны иметь значение.

**Условия целостности для отношения Категория:**

Номер категории уникальным образом идентифицирует кортеж в отношении, уникален для каждого кортежа.

Все атрибуты должны иметь значение.

**Условия целостности для отношения Издательство:**

Номер издательства уникальным образом идентифицирует кортеж в отношении, уникален для каждого кортежа.

Все атрибуты должны иметь значение.

**Условия целостности для отношения Тема:**

Номер темы уникальным образом идентифицирует кортеж в отношении, уникален для каждого кортежа.

Все атрибуты должны иметь значение.

**Условия целостности для отношения Книга:**

Номер книги уникальным образом идентифицирует кортеж в отношении, уникален для каждого кортежа.

Количество страниц должно быть больше 0.

Количество экземпляров должно быть больше 0.

Год издания должен быть больше 0.

Все атрибуты должны иметь значение.

### **2.10 Виртуальные отношения**

Виртуальное отношение – это отношение, образованное с помощью реляционных выражений на основе одного или нескольких отношений (определение представлений). Все представления (виртуальные отношения) должны автоматически обновляться при модификации данных в базовых отношениях.

В данной системе базы данных может часто потребоваться вывод определенной информации из отношений: Пожертвование, Кампания, Пользователь.

Эта информация нигде не хранится и высчитывается каждый раз заново на основе того сколько было совершенно операций с этими отношениями.

Для уменьшения затрат производительности системы базы данных следует с помощью реляционных выражений составим виртуальное отношение, содержащее основную информацию о необходимых для выполнения операции атрибутов.

Обновление соответствующего виртуального отношения будет производиться в случае осуществления операций с одним из отношений.

Реляционное выражение для вывода списка студентов:

σ (номер = `номер студента`) Students

Реляционное выражение для вывода списка карточек студентов:

σ (номер = `номер карточки студента`) SCards

Реляционное выражение для вывода списка книг:

σ (номер = `номер книги`) Books

### **2.11 Реляционные выражения**

Реляционное выражение - выражение над отношениями, составленное из реляционных операций. Реляционное выражение – тоже отношение.

Пользователь должен иметь возможность осуществлять следующие запросы:

* Создать карточку для студента  
  SCards = SCards ∪ {‘дата выдачи’, ‘дата окончания’, ’номер студента’, ’номер книги’, ’номер библиотеки’, ’номер библиотекаря’}
* Зарегистрировать студента   
  Students = Students ∪ {‘имя’, ‘фамилия’, ’номер группы’, ’номер семестра’}
* Добавить книгу  
  Books = Books ∪ {‘название’, ‘кол-во страниц’, ’год издания’, ’номер темы’, ’номер категории’, ’номер автора’, ’номер издательства’, ’номер библиотеки’, ’комментарий’ , ’кол-во экземпляров’}

#### Физическая реализация модели базы данных

В данной главе описывается реализация взаимодействия прикладного приложения с базой данных с использованием языка С# и Entity Framework.

Программная реализация запроса на создание библиотеки:

public void addLibrary(string name)

{

LibraryContext db = new LibraryContext();

Lib lib = new Lib(name);

db.Libs.Add(lib);

}

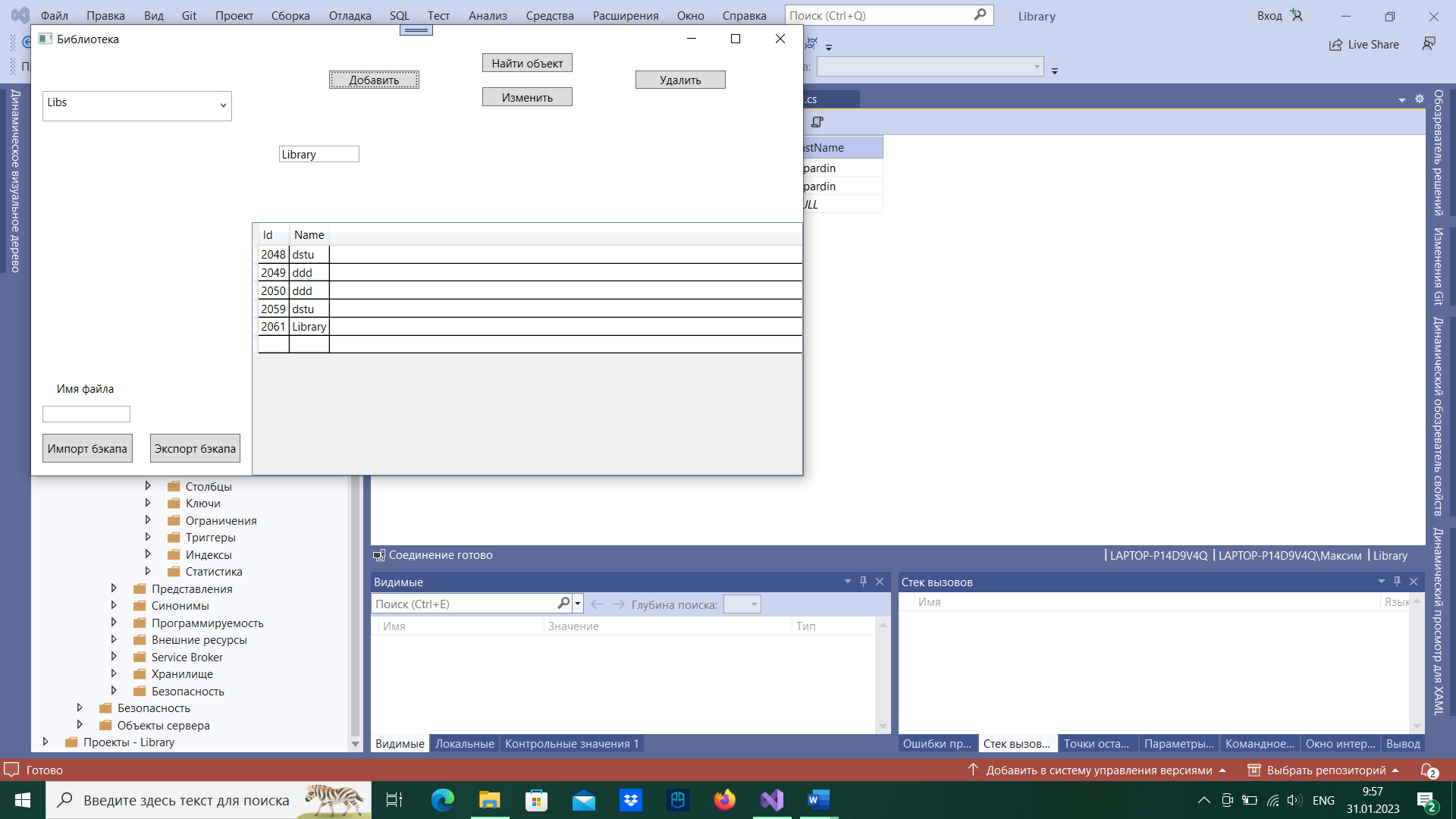


Рисунок 3.1 – представление создания библиотеки

Программная реализация запроса на удаление библиотеки:

public void deleteLibrary(int id)

{

LibraryContext db = new LibraryContext();

foreach (Lib lib in db.Libs)

if (lib.Id == id) db.Libs.Remove(lib);

}

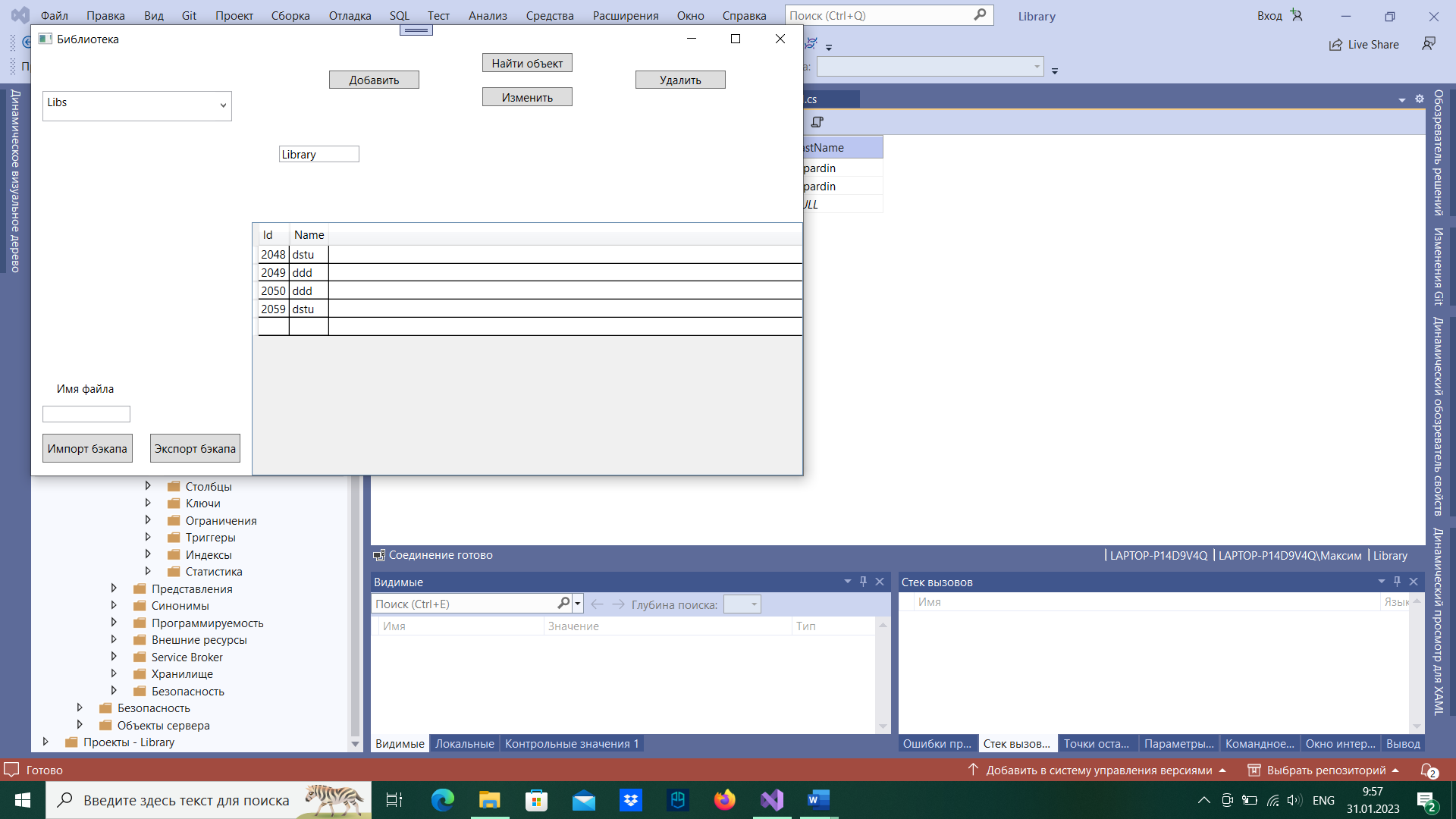


Рисунок 3.2 – представление удаления библиотеки

Программная реализация запроса на создание библиотекаря:

public void addLibrarian(string fname,string lname,int id\_lib)

{

LibraryContext db = new LibraryContext();

Librarian libr = new Librarian(fname,lname,id\_lib);

db.Librarians.Add(libr);

}

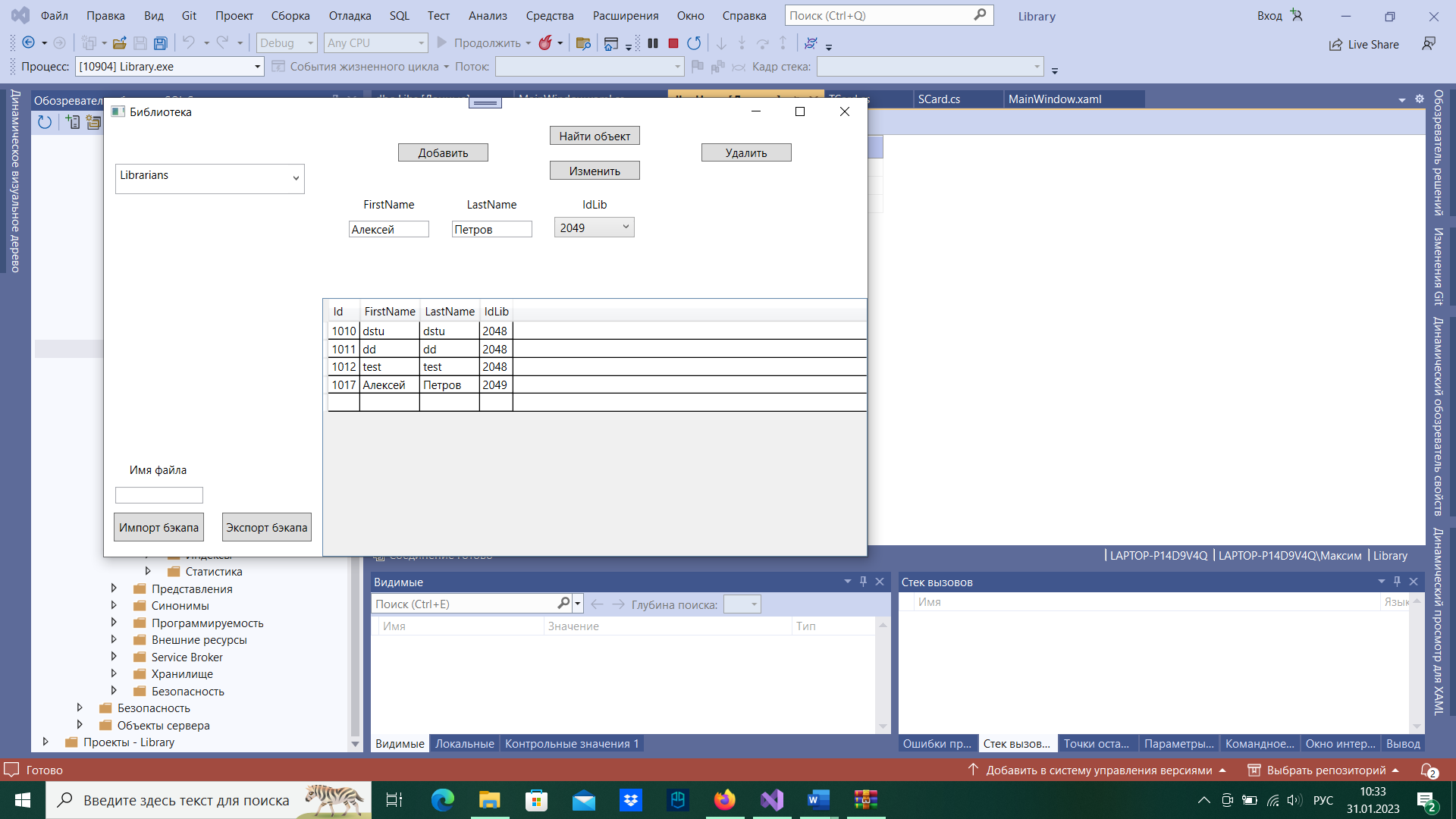


Рисунок 3.3 – представление создания библиотекаря

Программная реализация запроса на удаление библиотекаря:

public void deleteLibrarian(int id)

{

LibraryContext db = new LibraryContext();

foreach (Librarian libr in db.Librarians)

if (libr.Id == id) db.Librarians.Remove(libr);

}

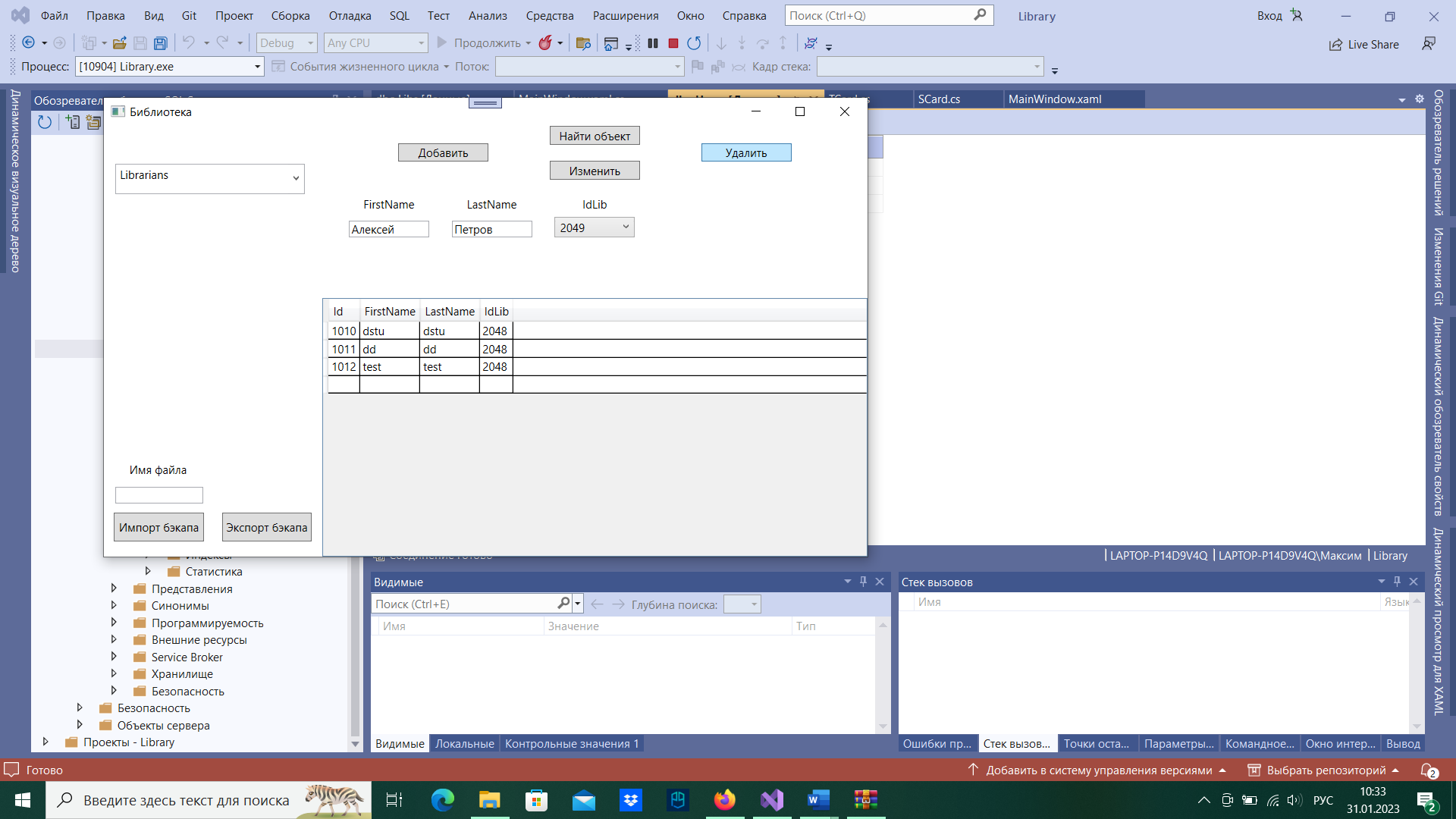


Рисунок 3.4 – представление удаления библиотекаря

Программная реализация запроса на создание книги:

public void addBook(string name,int pages,int yearpress,int id\_themes,

int id\_category,int id\_author, int id\_press,int id\_lib,string comment,int quantity)

{

LibraryContext db = new LibraryContext();

Book book = new Book(name,pages,yearpress,id\_themes,id\_category,

id\_author,id\_press,id\_lib,comment,quantity);

db.Books.Add(book);

}



Рисунок 3.5 – представление создания книги

Программная реализация запроса на удаление книги:

public void deleteBook(int id)

{

LibraryContext db = new LibraryContext();

foreach (Book book in db.Books)

if (book.Id == id) db.Books.Remove(book);

}

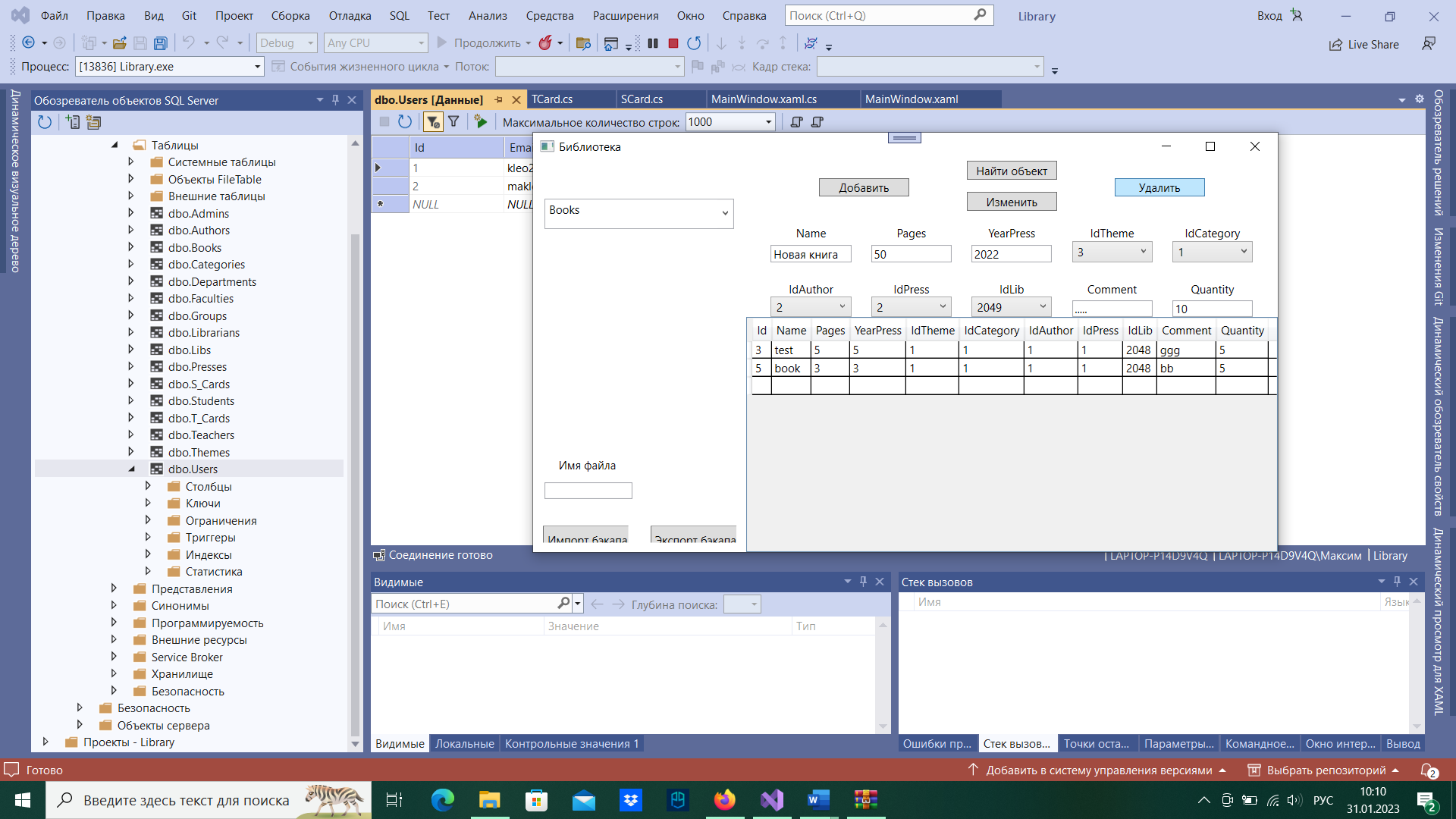


Рисунок 3.6 – представление удаления книги

Программная реализация запроса на создание карточки студента:

public void addSCard(int id\_student, int id\_book, int id\_lib,

int id\_libr, DateTime datein, DateTime dateout)

{

LibraryContext db = new LibraryContext();

SCard scard = new SCard(id\_student,id\_book,id\_lib,id\_libr,datein,dateout);

db.SCards.Add(scard);

}

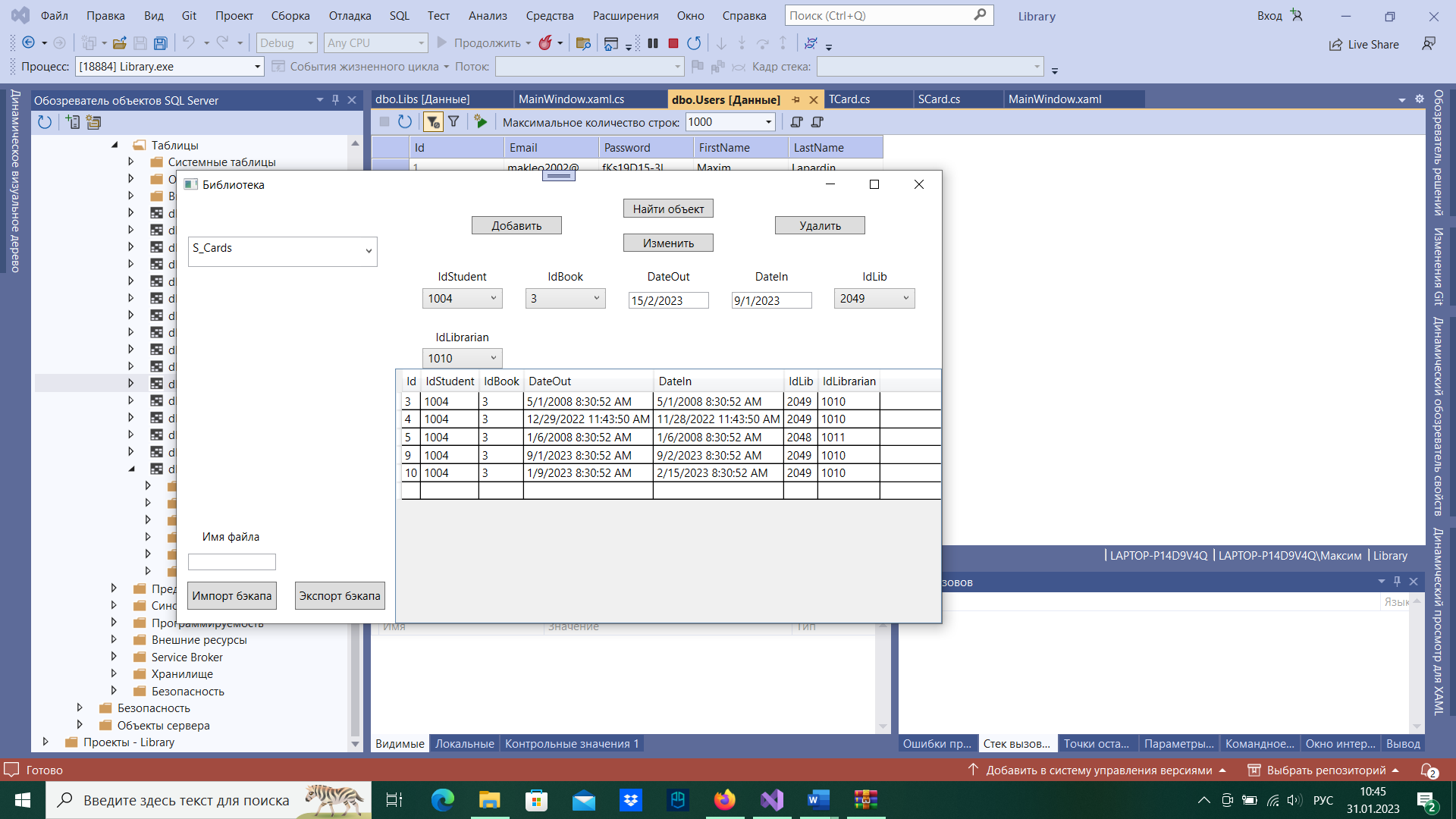


Рисунок 3.7 – представление создания карточки студента

Программная реализация запроса на удаление карточки студента:

public void deleteSCard(int id)

{

LibraryContext db = new LibraryContext();

foreach (SCard scard in db.SCards)

if (scard.Id == id) db.SCards.Remove(scard);

}

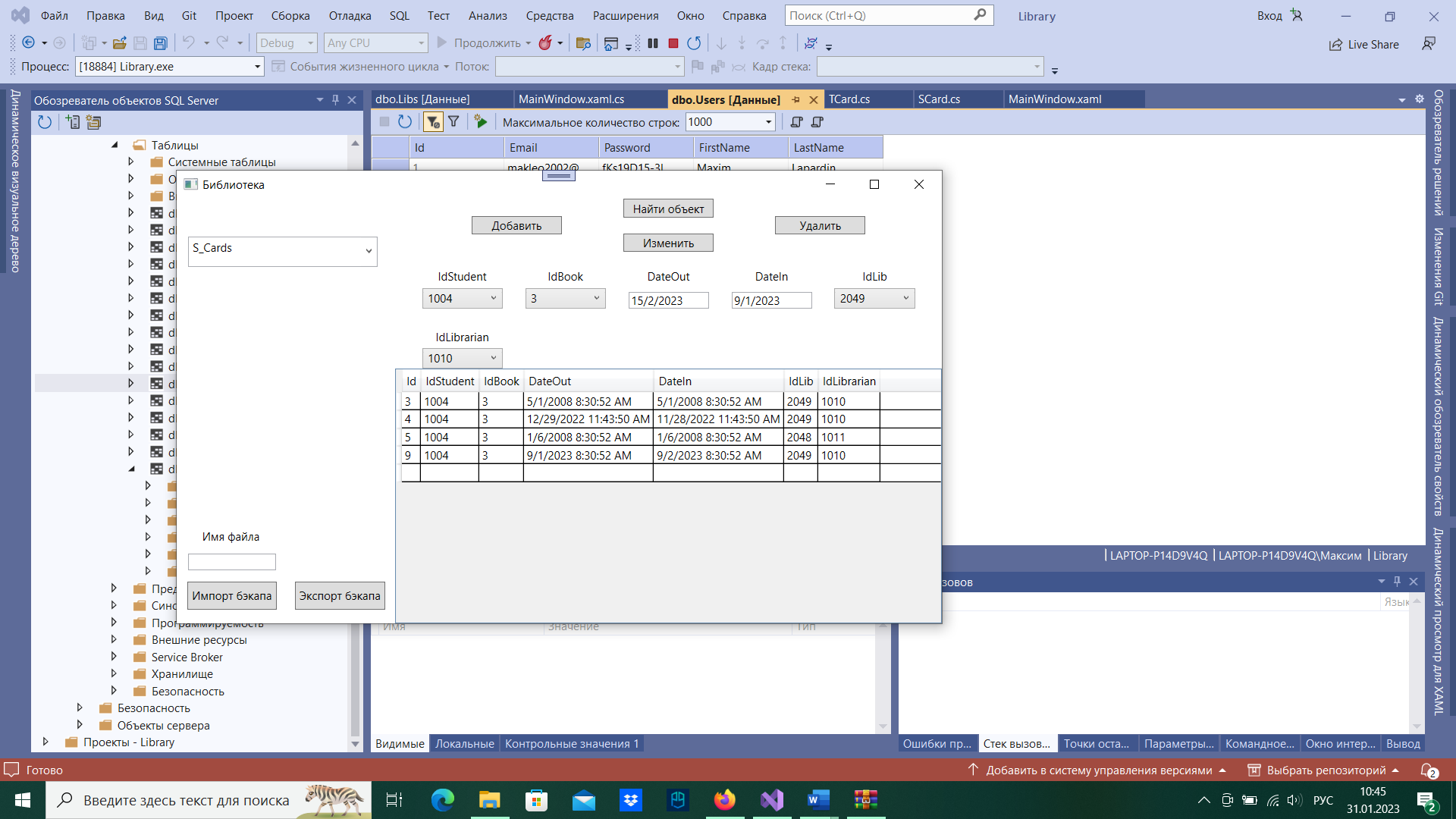


Рисунок 3.8 – представление удаления карточки студента

Программная реализация запроса на создание карточки учителя:

public void addTCard(int id\_teacher, int id\_book, int id\_lib,

int id\_libr, DateTime datein, DateTime dateout)

{

LibraryContext db = new LibraryContext();

TCard tcard = new TCard(id\_teacher, id\_book, datein, dateout, id\_lib, id\_libr);

db.TCards.Add(tcard);

}



Рисунок 3.9 – представление создания карточки учителя

Программная реализация запроса на удаление карточки учителя:

public void deleteTCard(int id)

{

LibraryContext db = new LibraryContext();

foreach (TCard tcard in db.TCards)

if (tcard.Id == id) db.TCards.Remove(tcard);

}

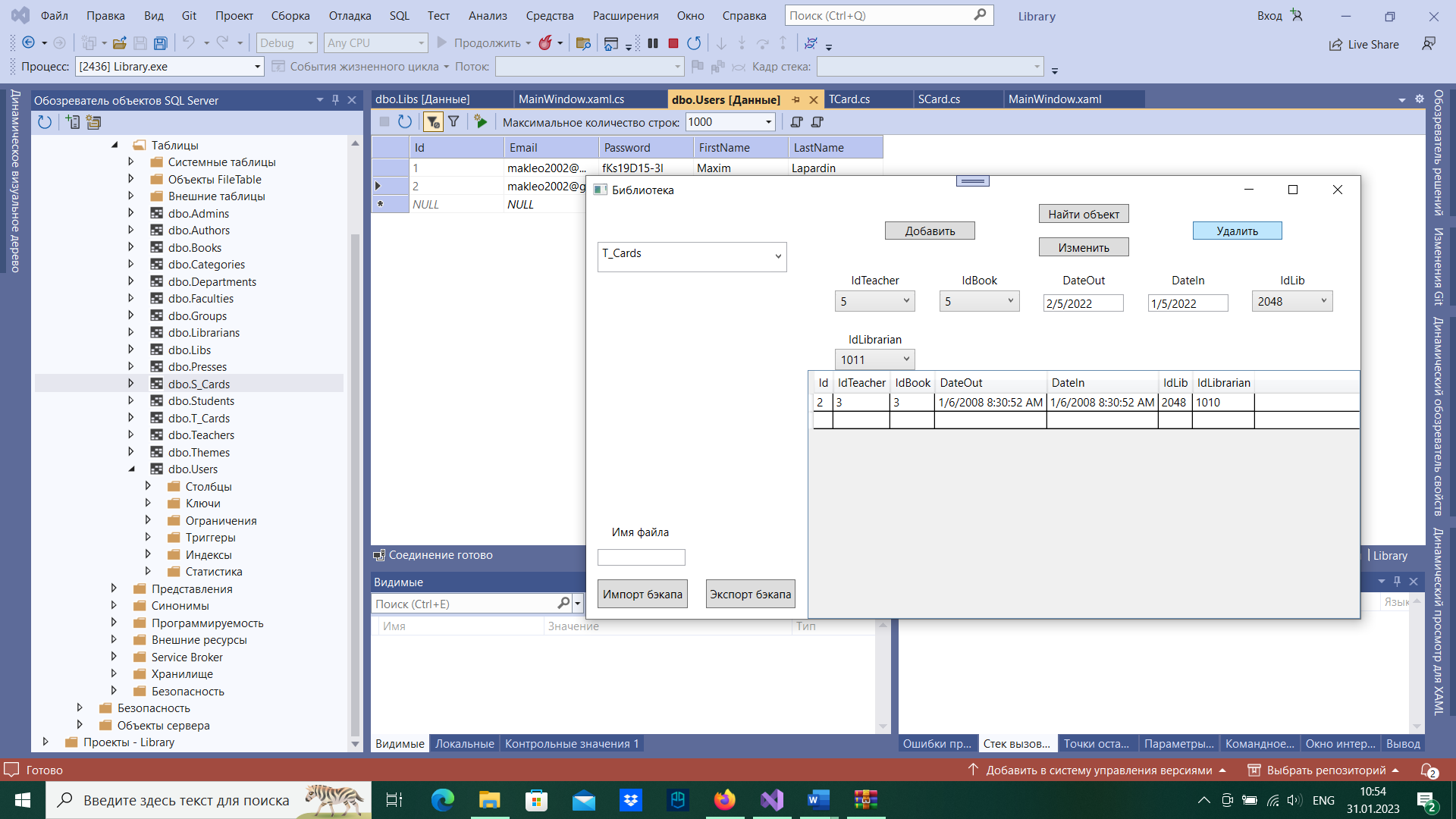


Рисунок 3.10 – представление удаления карточки учителя