**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»**

Навчально-науковий інститут атомної та теплової енергетики

Кафедра цифрових технологій в енергетиці

**КУРСОВА РОБОТА**

з дисципліни « Системи баз даних»

(назва дисципліни)

на тему: «Додаток для управління Базою Даних Бібліотеки»

|  |
| --- |
| Студента групи ТР-12 |
| зі спеціальності 122 – «Комп’ютерні науки» |
| Руденко В.І. |
| (Прізвище ініціали) |
| Керівник доцент, к.е.н., доцент Сегеда І.В. |
| (посада, вчене звання, науковий ступінь, прізвище та ініціали) |
| Керівник асистент Софієнко А.Ю. |
| (посада, вчене звання, науковий ступінь, прізвище та ініціали) |
| Кількість балів: \_\_\_\_\_Оцінка: \_\_\_\_\_\_\_\_\_\_\_ |
|  |

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (вчене звання, науковий ступінь, прізвище та ініціали

Київ - 2023 рік

**Національний технічний університет України**

**«Київський політехнічний інститут ім. Ігоря Сікорського»**

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ

Кафедра ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ

Рівень вищої освіти \_\_\_\_Бакалавр\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

За освітньою програмою «Цифрові технології в енергетиці»

Спеціальності 122 Комп’ютерні науки

**З А В Д А Н Н Я**

**НА КУРСОВУ РОБОТУ СТУДЕНТУ**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Руденка Владислава Ігоровича\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище, ім’я, по батькові)

1.Тема роботи «Додаток для управління Базою Даних Бібліотеки»\_\_\_\_\_\_\_\_\_\_\_\_\_\_

керівник курсової роботи - \_Сегеда Ірина Василівна, к.е.н, доцент\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

2. Строк подання студентом роботи 25 грудня 2023 року \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Вихідні дані до проекту (роботи): середовище розробки – MySQL Workbench, Windows Presentation Foundation, C#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) –принцип роботи графічного інтерфейсу та рисунки таблиць бази даних

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів виконання дипломного  проекту (роботи) | Строк виконання етапів проекту  ( роботи ) | Примітка |
| 1. | Вибір та затвердження теми роботи | 13.09.2023 |  |
| 2. | Вивчення предметної області | 16.09.2023 |  |
| 3. | Проектування бази даних | 15.10.2023 |  |
| 4. | Заповнення Бази-Даних проекту | 25.10.2023 |  |
| 5. | Розробка Графічного Інтерфейсу продукту | 01.11.2023 |  |
| 6. | Оформлення пояснювальної записки | 15.12.2023-20.12.2023 |  |

**Студент \_\_\_\_\_\_\_\_\_ Владислав РУДЕНКО**

( підпис ) (ім’я ПРИЗВИЩЕ)

**Керівник курсової роботи \_\_\_\_\_\_\_\_\_ Ірина СЕГЕДА**

( підпис ) (ім’я ПРИЗВИЩЕ)

**Керівник курсової роботи \_\_\_\_\_\_\_\_\_ Антон СОФІЄНКО**

( підпис ) (ім’я ПРИЗВИЩЕ)

**АНОТАЦІЯ**

Результатом роботи є програмний додаток з графічним інтерфейсом, що створений для взаємодії з Базою Даних Бібліотеки. Функціонал якого дозволяє створювати, видаляти та оперувати даними бази даних для досягнення ефективної роботи бібліотеки. Проводиться аналіз різних методів виконання поставленої задачі.

Для реалізації поставленої задачі розроблено Базу Даних, що спроможна зберігати та оперувати інформацією про бібліотеку, її клієнтах. Та імітувати, на цій основі, договори оренди.

Зручна взаємодія з користувачем досягається за рахунок графічного інтерфейсу, який розроблений на сучасній мові програмування С#. Для створення Бази Даних використано систему управління базами даних MySQL, що використовує мову запитів SQL

Курсова Робота містить теоретичні відомості згідно до понять принципу роботи та взаємодії бази даних, а також інформацію щодо розробки застосунку для керування системою бібліотеки.

Ключові слова: База Даних, MySQL, SQL, Бібліотека, графічний інтерфейс, запит, Мова програмування С#, СУБД

**ANNOTATION**

The result of the work is a software application with a graphical interface, which is created to interact with the Library Database. The functionality of which allows you to create, delete and operate database data to achieve efficient operation of the library. The analysis of various methods of implementation of the task.

To implement the task developed Database, which is able to store and operate information about the library, its customers. And simulate, on this basis, lease agreements.

Convenient interaction with the user is achieved through a graphical interface, which is developed in the modern C # programming language. To create the Database, the MySQL database management system was used, which uses the SQL query language

The course work contains theoretical information according to the concepts of the principle of operation and interaction of the database, as well as information on the development of an application for managing the library system.

Keywords: Database, MySQL, SQL, Library, graphical interface, query, C# programming language, DBMS

**ЗМІСТ**

ВСТУП…………………………………………………………………………………..7

1. АНАЛІЗ ВИМОГ ДОДАТКУ………………………………………………………8
   1. Постановка завдання………………………………………………………...8
   2. Розробка концептуальної моделі……………………………………………8
   3. Діаграма прецедентів……………………………………………………....10
2. ЗАСОБИ РОЗРОБКИ………………………………………………………….......11
   1. Система Управління Базами Даних MySQL………………………………11
   2. Інтегроване середовище Visual Studio………………………………….....11
   3. Графічна система Windows Presentation Foundation……………………...12
3. РОЗРОБКА БАЗИ ДАНИХ ……………………………………………………….13
   1. Створення таблиць…………………………………………………………13
   2. Створення тригерів…………………………………………………………22
   3. Створення процедур……………………………………………………......24
   4. Створення представлень…………………………………………………...28
4. РОЗРОБКА ДОДАТКУ…………………………………………………………...32
   1. Створення системи меню………………………………………………......32
   2. Представлення даних в застосунку………………………………………..34
   3. Взаємодія з елементами бази даних……………………………………….36
      1. Створення нових записів………….……………………………...36
      2. Редагування існуючих записів….……………………………......38
      3. Видалення даних………………………………………………….40

ВИСНОВКИ…………………………………………………………………………...42

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ…………………………………………......43

ДОДАТКИ…………………………………………………………………………......44

**ВСТУП**

Розвиток інформаційних технологій та їх поширення серед мас дало змогу створювати потужні та ефективні засоби управління різними областями життєдіяльності людини. Одним з найдавніших джерел отримання інформації є бібліотека і поліпшення систем її роботи є важливим фактором розвитку суспільства.

Метою роботи є створення застосунку для імітації керування діяльністю бібліотеки. Головним завданням є створення Бази Даних та Інтерфейсу, що допоможуть реалізувати систему моніторингу наявними екземплярами літератури та надати послуги оренди даними книгами. На основі цього головним завданням роботи є розробка бази даних, створення програмної частини для використання бази даних та забезпечення зручного інтерфейсу для взаємодії з користувачем.

Об’єктом дослідження є система взаємодії бібліотеки, а предметом дослідження – додаток, який дозволить ефективно керувати бібліотекою за рахунок використання баз даних.

Робота додатку передбачає створення , редагування, видалення існуючих екземплярів літератури бібліотеки. Функціонал надає можливість моніторингу за розташуванням даних об’єктів та забезпечує ефективну взаємодію з ними. Прикладом функціоналу є створення системи оренди, яка включає в себе інформацію про орендаря, об’єкт оренди, який представлений у вигляді екземпляру книги та дані договору, на час якого відбувається оренда.

1. АНАЛІЗ ВИМОГ
   1. Постановка завдання

Згідно до предметної області, додаток має містити функціонал для взаємодії з літературою бібліотеки. Включаючи в себе можливість перегляду усієї необхідної інформації, додавання нових книг, редагування існуючих книг та видалення інформації стосовно них. Для функціоналу оренди організувати можливість обрахунку ціни оренди в залежності від параметрів книги та скидки, якою володіє клієнт.

* 1. Розробка концептуальної моделі

Концептуальна модель для розробки є абстрактним представленням складних систем, програм або процесів, яке допомагає у розумінні їх основних складових частин, взаємозв'язків та принципів функціонування. Ця модель визначає ключові аспекти системи або програми на високому рівні, не вдаваючись в деталі чи конкретику реалізації.

На рисунку 1.1 можна ознайомитись з першою частиною концептуальної моделі додатку управління бібліотекою. Головною сутністю Бази Даних є Книга. Розробка її частин займає найважливіше місце, оскільки саме цей елемент є фундаментом взаємодій даної Бази Даних

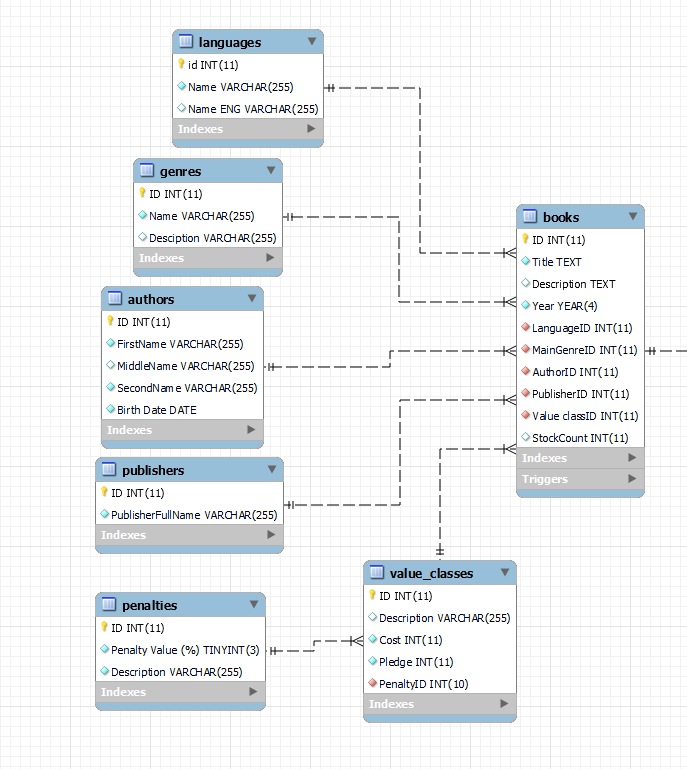


Рисунок 1.1 Концептуальна модель Бази Даних (Частина Перша)

Другою частиною Концептуальної модел є таблиці, які побудовані з метою надати необхідний функціонал до користувача та клієнта.(Рисунок 1.2)

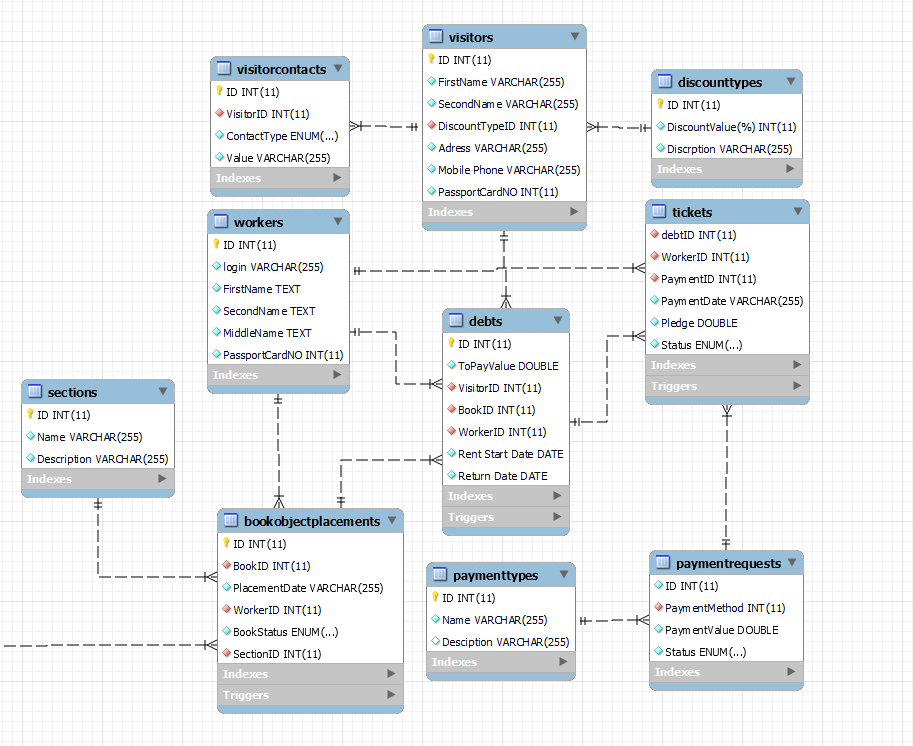


Рисунок 1.2 Концептуальна модель Бази Даних (Частина Перша)

Таблиця bookobjectplacement включає в себе зв’язок «Один до Багатьох» з таблицею books. Це дає змогу створювати екземпляри існуючих книг, з якими можливо оперувати в майбутньому.

* 1. Діаграма Прецедентів

Діаграма прецедентів - це представлення взаємодії користувача з системою, яке показує зв'язок між користувачем і різними випадками використання, в яких користувач бере участь. Прикладна діаграма використання UML є основною формою системних/програмних вимог для нової програми. Ключова концепція моделювання випадків використання полягає в тому, що вона допомагає нам проектувати систему з точки зору кінцевого користувача. Це ефективний метод зв'язку поведінки системи в термінах користувача, вказуючи всю зовнішню видиму поведінку системи.[1]

З розробленою Діаграмою Прецедентів згідно заданої предметної області можна ознайомитись на рисунку 1.3



Рисунок 1.3 Діаграма прецедентів

Єдиним актором даної системи є Бібліотекар, він відповідає за взаємодію з книгами, клієнтами та стовренням договорів оренди на окремі екземпляри книг.

1. ЗАСОБИ РОЗРОБКИ
   1. Система Управління Базами Даних MySQL

На сьогоднішній день існує багато систем управління базами даних. Але серед усіх є одна, що виділяється - MySQL. Це відкрита система управління базами даних, яка є дуже поширеною серед представників свого виду.

По-перше за все це відкрита та безкоштовна система управління базами даних з вражаючою швидкістю та ефективністю. MySQL здатна працювати працювати з великими, за обсягами, проектами та виконувати свою роботу без втрат продуктивності.

По-Друге ця система управління базами даних має великий перелік внутрішнього функціоналу. Що забезпечує її надійність та гнучкість. Маючи можливість конфігуруватись під потреби проекту – MySQL є найкращим варіантом для вирішення задач даної курсової роботи.

* 1. Інтегроване середовище Visual Studio

Серед найбільш поширених інтегрованих середовищ розробки Visual Studio є однією з найпопулярніших. Розроблене компанією Microsoft це середовище підтримує різні мови програмування та надає можливість створювати додатки з використанням різних технологій та платформ. Для курсової роботи це дає спроможність використовувати мову програмування C# та весь його функціонал максимально ефективно.

Вбудований функціонал Visual Studio для розробки дозволяє завчасно дізнаватись про проблеми роботу коду, а також надає всі переваги використання функціоналу системи контролю версій, дебагера та автоматичного доповнення коду.

Різноманітність плагінів та підтримки стороннього програмного забезпечення надає ряд переваг, такий як розширення функціоналу або пристосування середовища до різних потреб. Що робить Visual Studio гнучким та універсальним інструментом в розробці додатків.

* 1. Графічна система Windows Presentation Foundation

Графічна підсистема Windows Presentation Foundation (WPF) є однією з ключових технологій для розробки графічних інтерфейсів користувача (GUI) в операційній системі Windows. Основна концепція WPF базується на використанні векторної графіки, що дає можливість створювати візуальні елементи додатків з високою роздільною здатністю та масштабованістю. WPF використовує мову розмітки XAML, що дозволяє відокремлювати візуальну частину додатку від логічної, що спрощує роботу команди розробників та дизайнерів.

Технологія WPF має великий перелік вбудованих елементів взаємодії з користувачем, що дає спроможність створювати графічні інтерфейси з великою функціональністю. Технологія залишається популярною через свою потужність та гнучкість, що і сприяло її вибору для даного проекту.

1. РОЗРОБКА БАЗИ ДАНИХ
   1. Створення таблиць

Для коректної роботи програми та досягнення мети роботи було створено 17 та задіяно 17 таблиць. Для заповнення книги необхідними атрибутами, з них виділено 6 таблиць: Мова, Жанр, Автори, Видавництва, Клас Цінності книги та збиток за книгу.

Спочатку необхідно розглянути таблиці полів атребутів

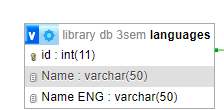


Рисунок 3.1

Таблиця «language» на рисунку 3.1 призначення для зберігання мови. Сама таблиця складається з 3-х полів:

* id - унікальний ідентифікатор (Primary Key) з встановленим значенням auto\_increment, це дозволяє в автоматичному порядку збільшувати значення ідентифікатора запису;
* Name - поле, що зберігає назву мови українською. Тип поля встановлений varchar з довжиною 50;
* ‘Name END’ –поле, що зберігає назву мови англійською. Встановлений тип поля varchar з довжиною 50;

Нижче на рисунку 3.2 представлено заповнену таблицю «language»

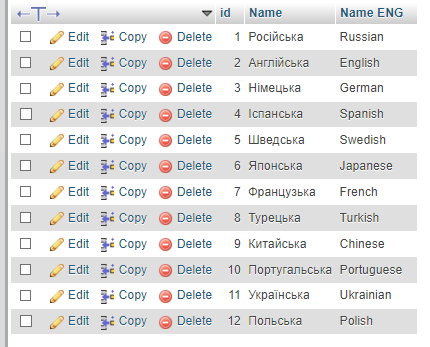


Рисунок 3.2 Заповнена таблиця «language»

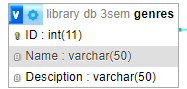
****

Рисунок 3.3 Таблиця «genres»

Таблиця «genres», що зображена на рисунку 3.3, зберігає перелік жанрів та їх опис. Складається з 3-х полів:

* id - унікальний ідентифікатор (Primary Key) з встановленим значенням auto\_increment, це дозволяє в автоматичному порядку збільшувати значення ідентифікатора запису;
* Name - поле, що зберігає назву мови Жанру.
* Description –поле, що зберігає Опис Жанру. Встановлений тип поля varchar з довжиною 50 та можливістю виставити значення null;

Нижче на рисунку 3.4 представлено заповенений вигляд таблиці «genres»



Рисунок 3.4 Заповнений вигляд таблиці «genres»

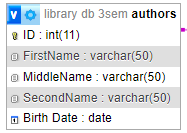


Рисунок 3.5 Таблиця authors

Представлена таблиця «authors» на рисунку 3.5 зберігає перелік інформації про авторів

Складається з 5-х полів:

* id - унікальний ідентифікатор (Primary Key)
* FirstName - поле, що містить Ім’я Автору.
* SecondName– поле, що містить Прізвище Автору. Встановлений тип поля varchar з довжиною 50 та можливістю виставити значення null;
* MiddleName – поле, що містить По-батькові Автору. Встановлений тип поля varchar з довжиною 50 та можливістю виставити значення null;
* Birth Date – поле містить дату народження.

Нижче на рисунку 3.6 представлено заповенений вигляд таблиці «authors»



Рисунок 3.6 Заповнений вигляд таблиці «authors»

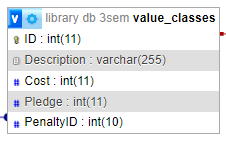


Рисунок 3.7 Таблиця «value\_class»

Таблиця «value\_class» зберігає перелік різних класів цінності книги, кожен з таких екземплярів має індивідуальні налаштування та вимоги до книги. Ознайомитись з нею можна на рисунку 3.7. Таблиця має зв’язок з таблицею «penalties» який відповідає типу «один-до-багатьох». Та складається з 5-х полів:

* id - це унікальний ідентифікатор (Primary Key)
* Description - поле, що містить Опис класу цінності книги.
* Cost–поле, що містить ціну Оренди книги.
* Pledge–поле, що містить оціночну вартість
* PenaltyID – поле-індекс, що містить зовнішній ключ (Foreign key) який посилається на поле ID таблиці «penalties». Це значення використовує таблицю penalties для отримання значення збитку, який буде нарахований в разі пошкодження книги

Далі створена таблиця «books» (Рисунок 3.8), що містить інформацію про книги, які можуть бути збережені та викладені бібліотекою. Таблиця має декілька зв’язків типу «один-до-багатьох» з такими таблицями як: «languages», «genres», «authors», «publishers», «value\_classes» .

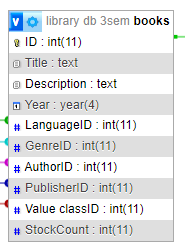


Рисунок 3.8 Таблиця «books»

Таблиця «books» містить 10 полів:

* id - унікальний ідентифікатор (Primary Key)
* Title –поле що містить назву книги;
* Description –поле, що містить інформацію про опис книги
* Year – поле, яке зберігає дату видання
* LanguageID –поле-індекс, що містить зовнішній ключ (Foreign key) який посилається на поле ID таблиці «penalties», значення таблиці необхідне щоб зберігати мову літератури
* GenreID - поле-індекс, що містить зовнішній ключ (Foreign key) який посилається на поле ID таблиці «genres», служить для збереження значення жанру
* AuthorID - поле-індекс, що містить зовнішній ключ (Foreign key) який посилається на поле ID таблиці «authors», використовується для зберігання Прізвища Ім’я По-Батькові автора.
* PublisherID - поле-індекс, що містить зовнішній ключ (Foreign key) який посилається на поле ID таблиці «publishers», необхідний для отримання значення видавництва.
* `Value ClassID` - поле-індекс, що містить зовнішній ключ (Foreign key) який посилається на поле ID таблиці values\_class для отримання значень цін за книгу.
* StockCount – це значення типу int, яке зберігає кількість наявних книг цього типу

Нижче на рисунку 3.9 представлено заповенений вигляд таблиці «books»



Рисунок 3.9 Заповнений вигляд таблиці «books»

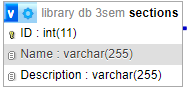


Рисунок 3.10 Таблиця «sections»

Створена таблиця на рисунку 3.10 це «sections» - дозволяє зберігати розташування книг, вона включає в себе 3 поля:

* id - унікальний ідентифікатор (Primary Key)
* Name - поле, що зберігає назву секції збереження
* Description –поле, що зберігає Опис Секції Збереження. Це можуть бути як особливі прикмети секції так і опис маршруту

Нижче на рисунку 3.11 представлено заповенений вигляд таблиці «sections»

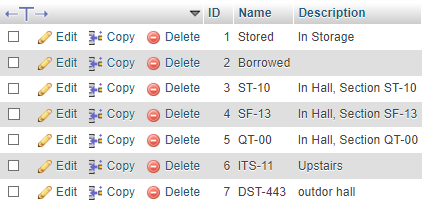


Рисунок 3.11 Заповнений вигляд таблиці «sections»

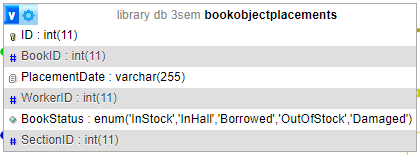


Рисунок 3.12 Таблиця «bookobjectplacements»

Таблиця «bookobjectplacements», яка представлена на рисунку 3.12, зберігає інформацію про об’єкти книг, що можуть бути задіяними при створені договору оренди. При створені було задіяно тип зв’язків «Один-до-багатьох». Таблиця включає в себе 6 полів:

* id - унікальний ідентифікатор (Primary Key)
* BookID- поле-індекс, що містить зовнішній ключ (Foreign key) який посилається на поле ID таблиці books, необхідний для зберігання відомостей згідно виставленої книги
* PlacementDate – поле, яке зберігає дату розміщення екземпляру книги
* WoorkerID - поле-індекс, що містить зовнішній ключ (Foreign key) який посилається на поле ID таблиці «workers» для моніторингу взаємодій робітників.
* BookStatus – поле з типом даних ‘enum’, зберігає 5 стани книги: «InStock», «InHall», «Borrowed», «OutOfStock», «Damaged»

Нижче на рисунку 3.13 представлено заповенений вигляд таблиці «bookobjectplacement»

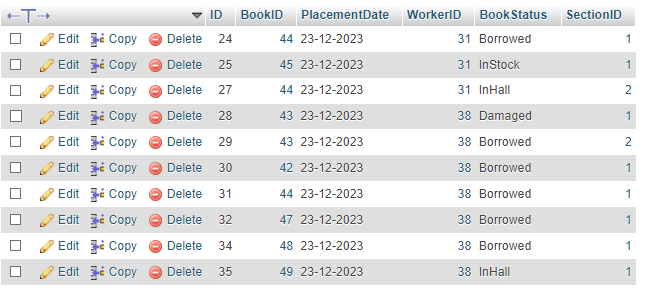


Рисунок 3.13 Заповнений вигляд таблиці «bookobjectplacements»

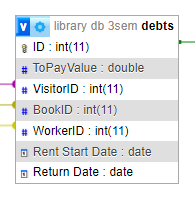


Рисунок 3.14 Таблиця «debts»

Мета таблиці на рисунку 3.14 – містити дані про створені договори оренди, об'єкт оренди, відвідувача. Таблиця має зв'язки «Один-до-Багатьох» та включає в себе 7 полів:

* id - унікальний ідентифікатор (Primary Key)
* ToPayValue – поле, що містить інформацію про суму до сплати за період оренди книги. Використаний тип даних double
* VisitorID - поле-індекс, що містить зовнішній ключ (Foreign key) який посилається на поле ID таблиці visistors, для відслідковування Клієнта
* BookID - поле-індекс, що містить зовнішній ключ (Foreign key) який посилається на поле ID таблиці books, для зберігання інформації про позичену книгу
* WorkerID - поле-індекс, що містить зовнішній ключ (Foreign key) який посилається на поле ID таблиці «workers», значення необхідне для наступного етапу моніторингу за персоналом
* `Rent Start Date` - поле, що зберігає інформацію стосовно початку часу оренди. Містить Тип даних date
* `Return Date` - поле, що містить час повернення книги. Тип даних date

Нижче на рисунку 3.15 представлено заповенений вигляд таблиці «debts»

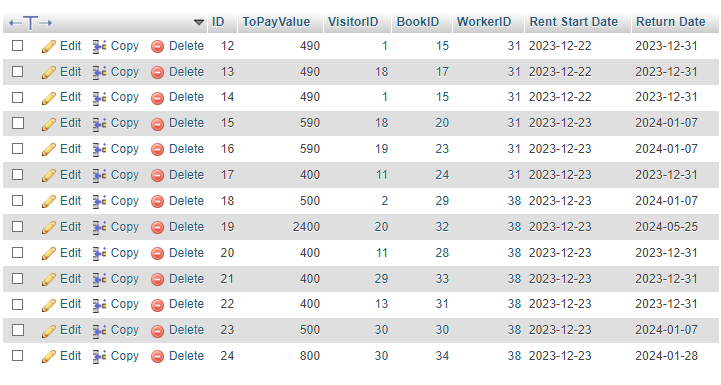


Рисунок 3.15 Заповнений вигляд таблиці «debts»

Останьою представленою таблицею є «visitors».Стоврення поля «visitors» можна побачити на рисунку 3.16

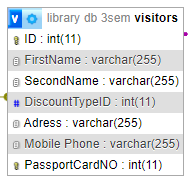


Рисунок 3.16

Таблиця «visitors» слугує колекцією клієнтів та їх інформації.. Таблиця вміщає в собі 7 полів:

* id - унікальний ідентифікатор (Primary Key)
* FirstName – слугує одиницею для зберігання Імені Клієнта. Має тип varchar з розміром 255.
* SecondName – аналогічно до «FirstName» зберігає інформацію про прізвище клієнта та має тип varchar розмірністю 255
* DyscountTypeID - поле-індекс, що мість зовнішній ключ (Foreign key) який посилається на поле ID таблиці «discounttypes», слугує для моніторингу статусу знижки на літературу.
* Adress – зберігає інформацію щодо місця проживання клієнта
* `Mobile Phone` - змінна для збереження базового контакта з клієнтом
* PassportCardNO – є номером паспорта клієнта, слугує для коректності роботи системи при клієнтах з однаковими іменами та прізвищами.

На відвідувача діє тригер з таблиці «tickets», який при кожному десятій закритій оренді збільшує тип скидки через зміну значень поля «DyscountTypeID».

* 1. Створення тригерів

Системи управління базами даних (СУБД) є ключовим елементом сучасних інформаційних технологій, що забезпечують зберігання, організацію та доступ до великих обсягів даних. У цьому контексті, виникає необхідність впровадження інструментів, що автоматизують обробку та контроль за цілісністю інформації. Одним з таких інструментів є тригери, що використовуються для автоматичної реакції на певні події в базі даних.

В даній роботі було реалізовано тригери для виконання різних задач. Але здебільшого вони використовуються для перевірки коректності оперуючих значень для забезпечення максимальної надійності при виконанні запитів.

На рисунку 3.17 Зображено створення тригеру, єдина задача якого перевіряти, чи книга, яку додають до оренди, доступна на складі. Якщо книга не відповідає вимогам тригер не дасть завершити запит і надішле системі управління базами даних помилку з текстом, що неможливо використати для оренди книжку, яка не в наявності.

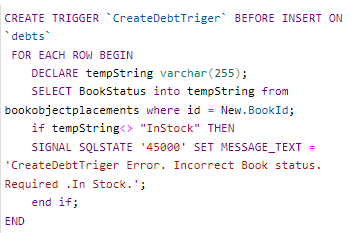


Рисунок 3.17 Тригер Перевірки наявності книги

Прості функціонали тригера забезпечують їх надійне виконання, тому функціонал використаних тригерів не містить об’ємних обчислень. Прикладом такого функціоналу є тригер обрахунку нової знижки, який зображений на рисунку 3.18. Це тригер оренди, який викликається при кожному оновлені рядка. Його функціонал включає в себе дві дії: перевірка чи новий статус оренди дорівнює завершеному, якщо це так – відбувається виклик процедури обрахунку знижки з передачою до нього параметру Відвідувача, за яким було закріплено даний договір.

Також для таблиці оренди було створено тригер, який при закриті транзакції, або її початку, виставляє відповідний статус для книги, яку позичають. Після чого та відмічається як «взята», що продемонстровано на рисунку

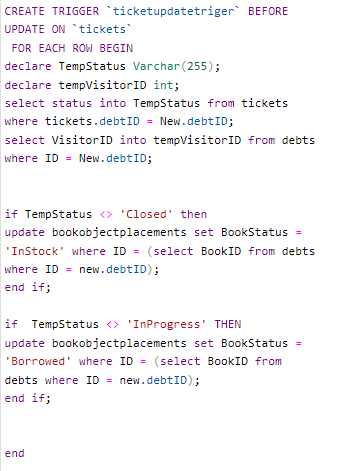


Рисунок 3.18 Тригер Оновлення статусу позиченої книги

Тригери можуть бути використані не тільки для обробки існуючої інформації, а й для створення нових об’єктів таблиць. Так, на рисунку 3.19 можна побачити, що для таблиці paymentrequests при додаванні нового елементу до таблиці debts за допомогою тригера створюється і новий екземпляр об’єкту для заданої таблиці .

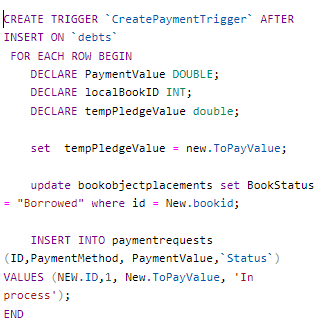


Рисунок 3.19 Тригер створення нового об’єкту до таблиці

Інші тригери Курсової роботи служать для корекції введеної інформації. Прикладом того є представлений на рисунку 3.20 тригер. Його мета – при створенні об’єкта перевірити, чи відповідають поля деяким умовам, якщо умова не задовольняється – відбувається заміна значень на стандартні, тобто нулю.

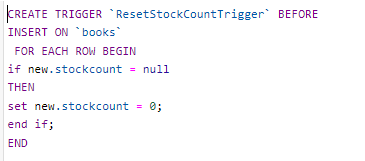


Рисунок 3.20 Тригер корекції введеної інформації

* 1. Створення процедур

Процедури в MySQL - це набір SQL-операторів, які зберігаються у базі даних. Вони можуть містити логіку, умовні конструкції, цикли та операції з даними. Їх основна перевага полягає в тому, що вони дозволяють створювати складніше логічне розгалуження та автоматизувати часті операції, що спрощує роботу з базою даних.

В курсовій роботі процедури займають одне з ключових місць. Один із головних елементів взаємодії з користувачем, а саме пошук – був представлений у вигляді процедури та зображений на рисунку 3.21



Рисунок 3.21 Зображення процедури пошуку часткових співпадінь

Хоч і дана процедура не має складної моделі, але за рахунок її простоти та продуманості вона вносить неабиякий внесок на загальну працездатність додатку. Процедура SearchSimilarByColumn отримує три параметри: назву таблиці, назву стовпця та значення для пошуку. Вона повертає користувачеві всі записи, де частина шуканого слова збігається за допомогою оператора LIKE. Символ % грає важливу роль у цьому пошуку, оскільки він підставляється для знаходження співпадінь, в залежності від розташування

Поставивши % спереду та позаду шуканого значення, оператор буде знаходити всі входження шуканого слова у записах та повертати їх. Дана процедура використовується в додатку для пошуку існуючих елементів книг за допомогою обраних параметрів пошуку, такі як: пошук за автором, жанром, назвою.



Рисунок 3.22 Зображення процедури пошуку повних співпадінь

У попередньої процедури існує і інша варіація: SearchbyColumn де так само існує три вхідні параметри: Назва Таблиці, Ім’я стовпця та шукане значення, але на відміну від процедури часткових співпадінь, дана – не має спец символу `%` оскільки її мета пошук точних співпадінь та повернення цих значень. Це використовується при пошуку конкретного користувача, оскільки кожен з них має унікальне значення поля PassportNO що зберігає номер паспорта.

Для обрахунку вартості оренди в залежності від терміну було створено спеціальну процедуру(рисунок 3.23), яка має вхідний параметр в якості розташованого екземпляру книги.



Рисунок 3.23 Процедура пошуку вартості книги

За допомогою простих запитів дана процедура зберігає та повертає 3 параметри: Ціну за тиждень оренди, Заставу та відсоток збитку, який буде взятий з застави у разі пошкодження книги. Варто зауважити що параметр відсотка збитку повертається у вигляді обрахованого значення, який залежить від значення Застави.

Також в проекті представлена процедура обрахунку значення знижки для покупця, яка зображена на рисунку 3.24



Рисунок 3.24 Обрахунок знижки для користувача

Процедура UpdateDiscountType викликається тригером після завершення договору та успішного повернення книги. Процедура вхідним параметром отримує ID відвідувача. Після чого відбувається обрахунок кількості завершених операцій та опираючись на результат обрахунку нараховує новий тип знижки відвідувачу. Якщо знижка що очікується більше ніж максимально існуюча – виставляється максимально можлива знижка .

Для створення нових працівників бібліотеки було спеціально створено процедуру, яка відповідає за створення юзерів та внесення їх до бази даних.

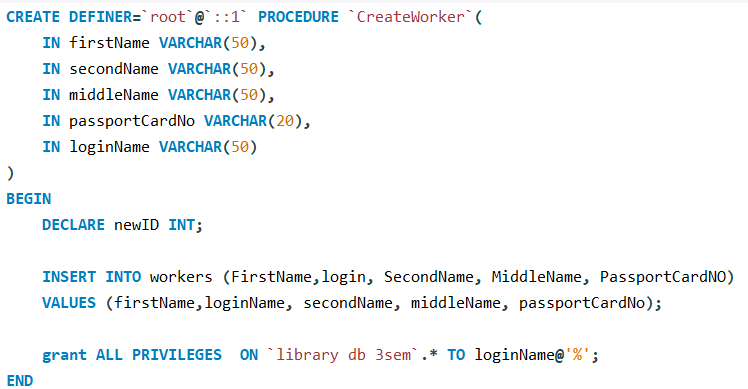


Рисунок 3.25 Процедура створення нового працівника

Вхідним параметром процедура приймає параметри особи, такі як Ім’я, Прізвище, По-батькові та номер паспорту для унікальності. Після чого створюється новий профіль користувача в залежності від введеного логіну.

* 1. Створення представлень

Представлення у системах управління базами даних є важливим інструментом для структурування та організації інформації. Вони визначають спосіб відображення даних користувачам або програмам, що сприяє зручності доступу до інформації.

В роботі було використано різного типу представлення для поліпшення взаємодії з користувачем, дані представлення здебільшого являються об’єктами замінниками, які Замінюють відображення значень ID до принятого для користувача вигляду.

Одним з найчастіше використовуваним представленням в даній роботі є представлення «bookobjectplacementview», що зображене на рисунку 3.26



Рисунок 3.26 Представлення існуючого переліку книжок

Це представлення створено для відображення всіх наявних книг у бібліотеці, щоб користувач мав можливість не лише відстежувати їх розташування, але й вносити зміни до їхніх даних за необхідності.

З метою поліпшення взаємодії з користувачем під час оформлення договору оренди, було створено представлення, яке відображає лише ті книги, які доступні для оренди, тобто ті, які є в наявності на складі, що представлено на рисунку 3.27.



Рисунок 3.27 Представлення доступних до оренди книжок

Створені представлення призначені для поліпшення взаємодії з іншими елементами відображення, зокрема, для спрощення взаємодії з договорами оренди, книжками та додатковими даними відвідувачів.



Рисунок 3.28 Представлення Існуючих Договорів Оренди

Відображення на рисунку 3.28 існуючих договорів оренди надає можливість оперативно відстежувати та, при необхідності, закривати окремі договори з оренди швидко та зручно

.

Рисунок 3.29 Представлення існуючих книжок

Віртуальна таблиця, зображена на малюнку 3.29, призначена для перетворення усіх ідентифікаторів (ID) на більш зрозумілі та зручні для користувача значення

1. РОЗРОБКА ГРАФІЧНОГО ІНТЕРФЕЙСУ
   1. Створення системи меню

Створений додаток призначений для взаємодії з базою даною з користувачем виключно у вигляді бібліотекаря. Графічний інтерфейс надає користувачеві доступ до перегляду існуючих даних, редагування елементів та створення нових екземплярів книг або записів стосовно відвідувачів.

Для того щоб мати доступ до функціоналу додатку спершу користувачеві необхідно авторизуватись за допомогою його власного логіну та паролю. Тому Головною сторінкою можна вважати меню авторизації, що зображене на рисунку.

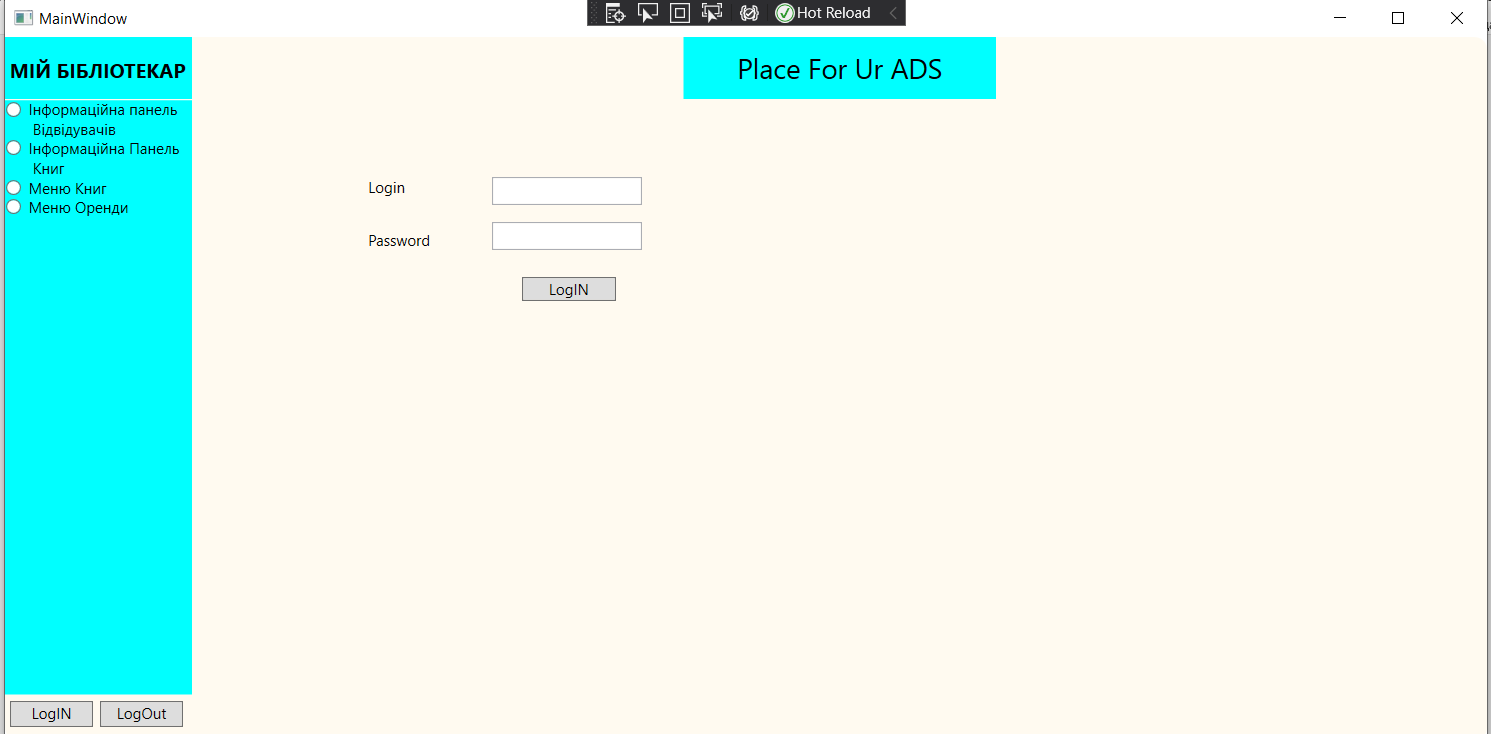


Рисунок 4.1 Головне меню графічного інтерфейсу

Після успішного введення паролю система повідомить про успішну авторизацію, що зображено на рисунку. Це означає, що тепер можливо скористатись функціоналом додатку.

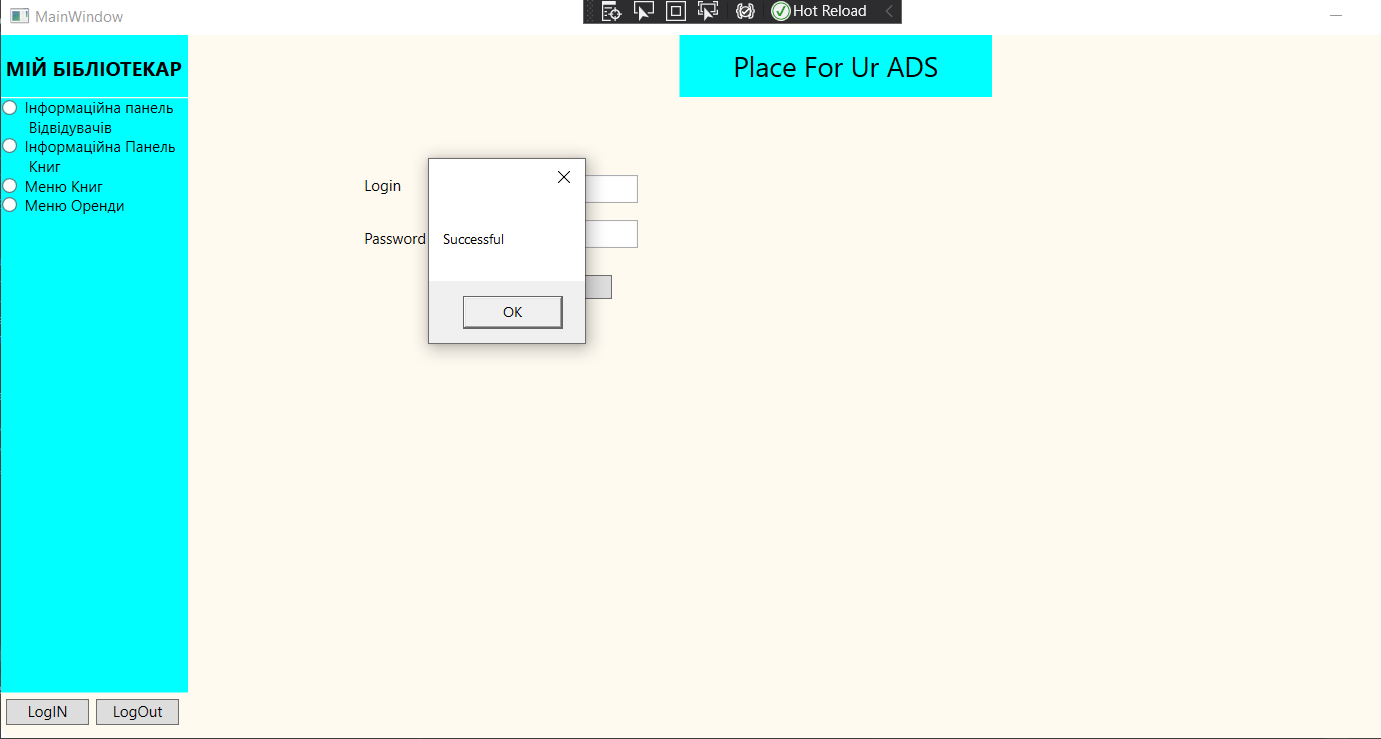
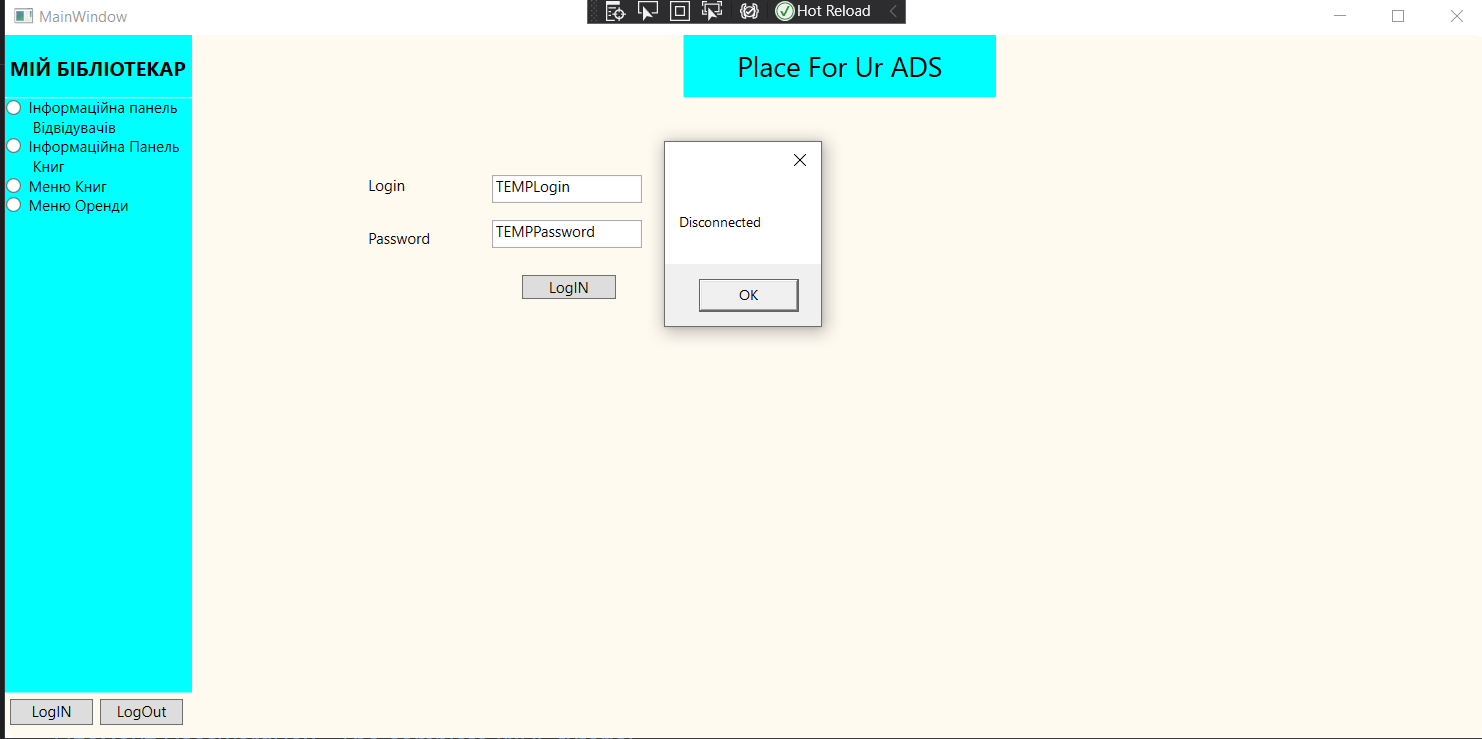


Рисунок 4.2 Повідомлення про успішну авторизацію

Варто зазначити що при хибно введених значенях скористатись функціоналом додатку, це можна наглядно побачити на рисунку

Рисунок 4.3 Відключення від системи при некоректних вхідних значеннях

Після авторизації користувачу доступні 4 меню на вибір, кожне з яких має свій власний функціонал та різну мету. Таким чином доступне меню поділяється на:

* Інформаційна Панель відвідувача - це інструмент для слідкування за даними про відвідувачів. Цей розділ надає можливість змінювати та, за потреби, видаляти інформацію.
* Інформаційна панель Книг – це розділ, призначений для слідкування за доступною для операцій інформацією. Враховуючи, що бібліотекарі не повинні мати права на редагування властивостей книги, функціонал для їх зміни не передбачено.
* Меню книг - розділ створений для додавання екземплярів літератури на основі наявних атрибутів. Кожен запис може бути змінений або видалений. Для полегшення взаємодії присутня система пошуку за атрибутами.
* Меню Оренди – Спеціальне меню надає бібліотекарю можливість оформляти договори оренди з користувачем. За допомогою доступних книг можливо обрати конкретну книгу та прив'язати її до облікового запису відвідувача. Якщо він згоден на операцію та після відображення його вартості, додатковий запис автоматично додається до таблиці "tickets" в цьому ж вікні. Статус оренди (tickets) може змінюватися, наприклад, при завершенні договору, коли відвідувач повертає книгу, бібліотекар може встановити статус договору як "Завершено", що автоматично відобразить цю інформацію та поверне книгу до переліку доступних для оренди.
  1. Представлення даних в застосунку

Для забезпечення максимальної ефективності та простоти виконання було вирішено відобразити значення у вигляді таблиці. Таким чином у розділі «Інформаційна панель» для відвідувачів та книг представлено різного виду таблиці з можливістю їх змін, що представлено на рисунку 1.1

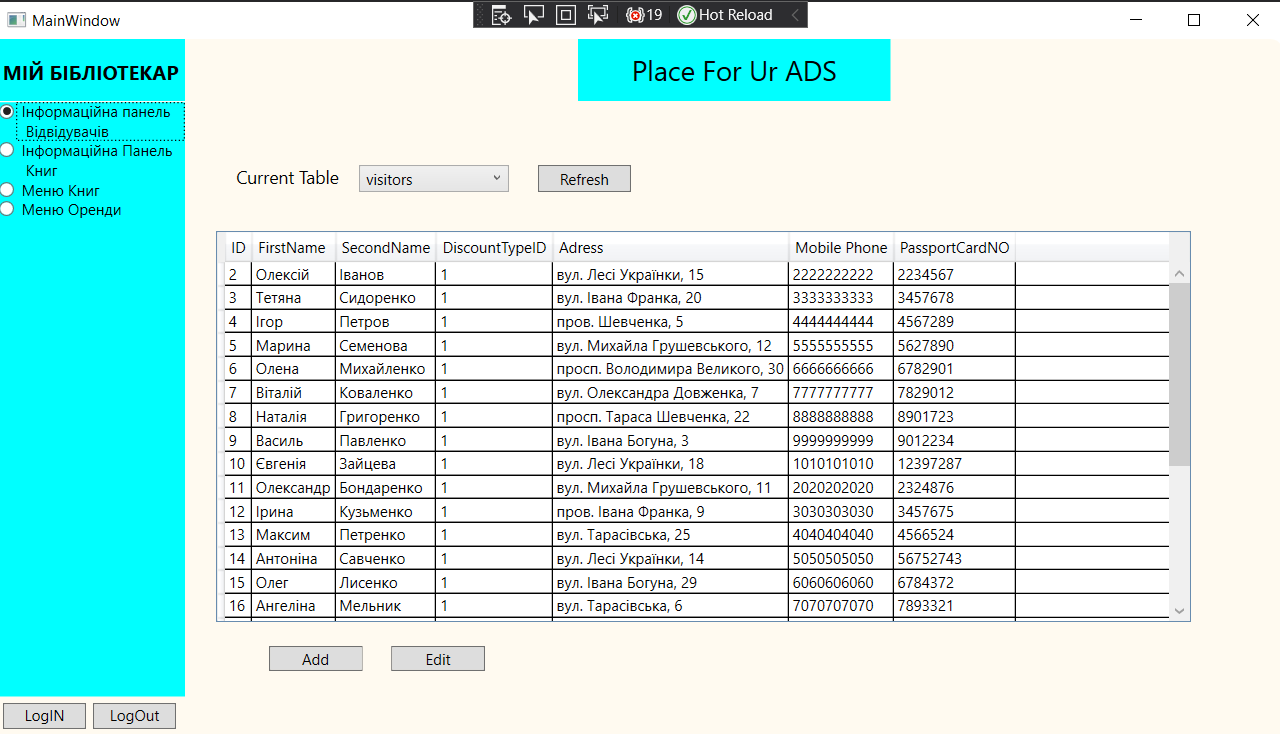


Рисунок 4.4 Відображення інформації у вигляді таблиць

Вигляд таблиць розділів Меню книг та Меню Оренди, що зображені на рисунку, відрізняються лише тим, що в них наявні елементи пошуку та прямої взаємодії з полями.

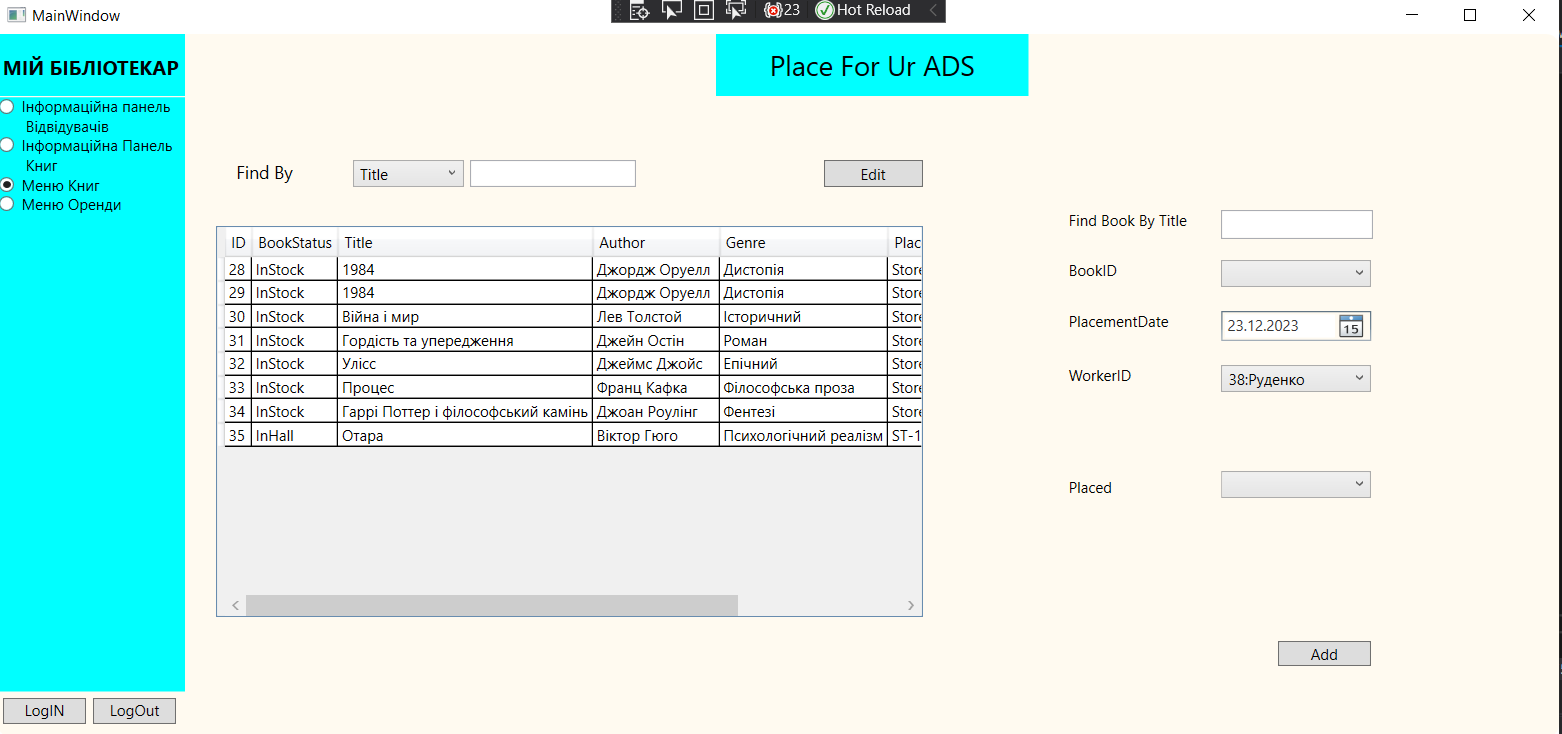


Рисунок 4.5 Відображення таблиць у Меню Книг

Таким чином бібліотекарю надається повний спектр інформації для подальшої взаємодії у вигляді додавання екземплярів книг, або створення договорів оренди.

* 1. Взаємодія з елементами бази даних
     1. Створення нових даних

Створення нових записів даних є невід’ємною частиною функціоналу графічного інтерфейсу. Додавання нових даних відбувається за рахунок взаємодії користувача з інтерфейсом, де необхідно власноруч обрати елементи до створення. На рисунку 1.1 Представлено методику додавання нового елементу до таблиці.

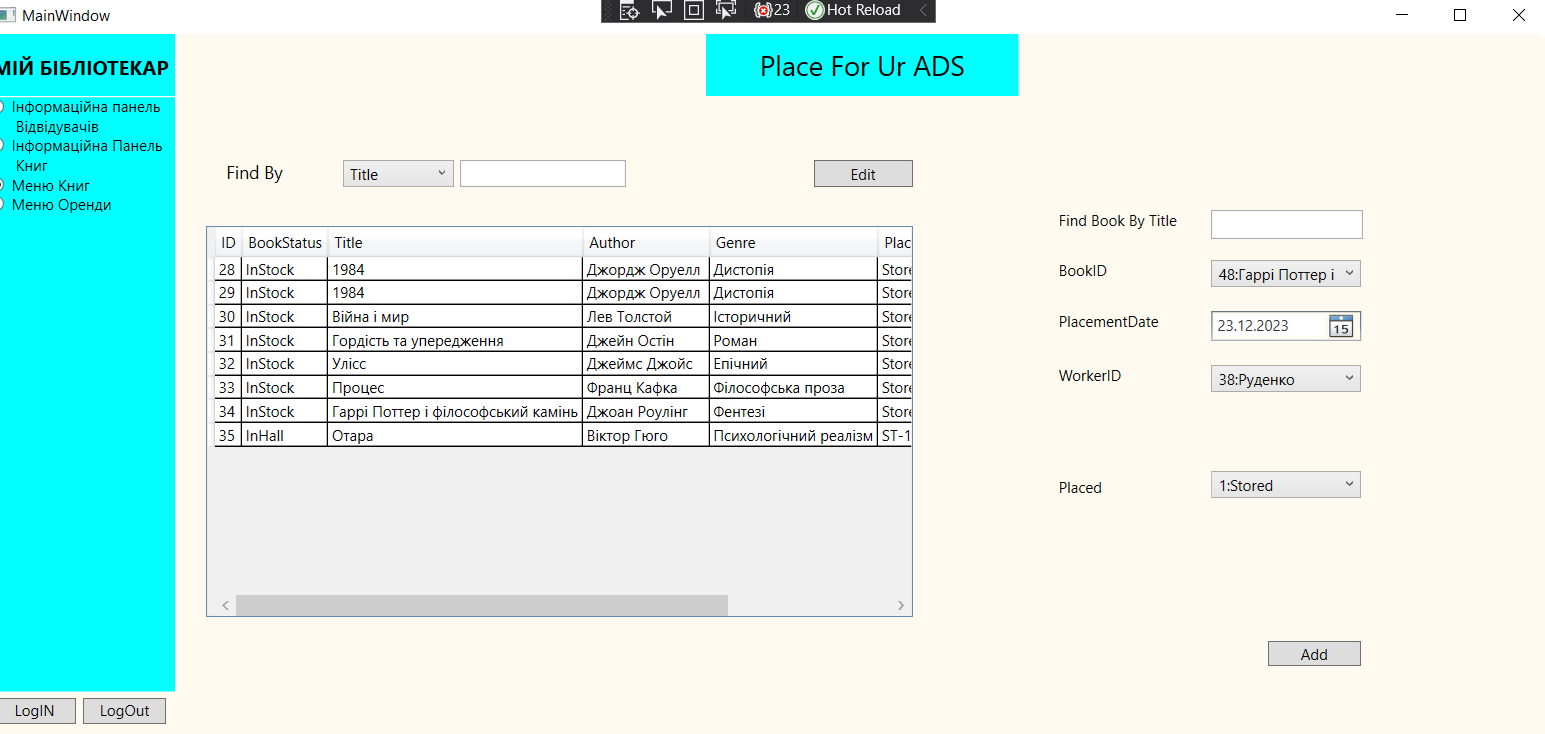


Рисунок 4.6 Створення нового запису до таблиці Книг

Для Меню книг необхідно обрати книгу зі списку, або знайти відповідну за допомогою функціоналу пошуку та обрати нове розташування книги. Це може бути як розміщення книги в залі (обираючи номер секції), або помітити книгу як доступною на складі, що дозволить створити на її основі договір оренди.

Меню Оренди відрізняється дещо іншим функціоналом створення запису, що можна побачити на рисунку 1.1

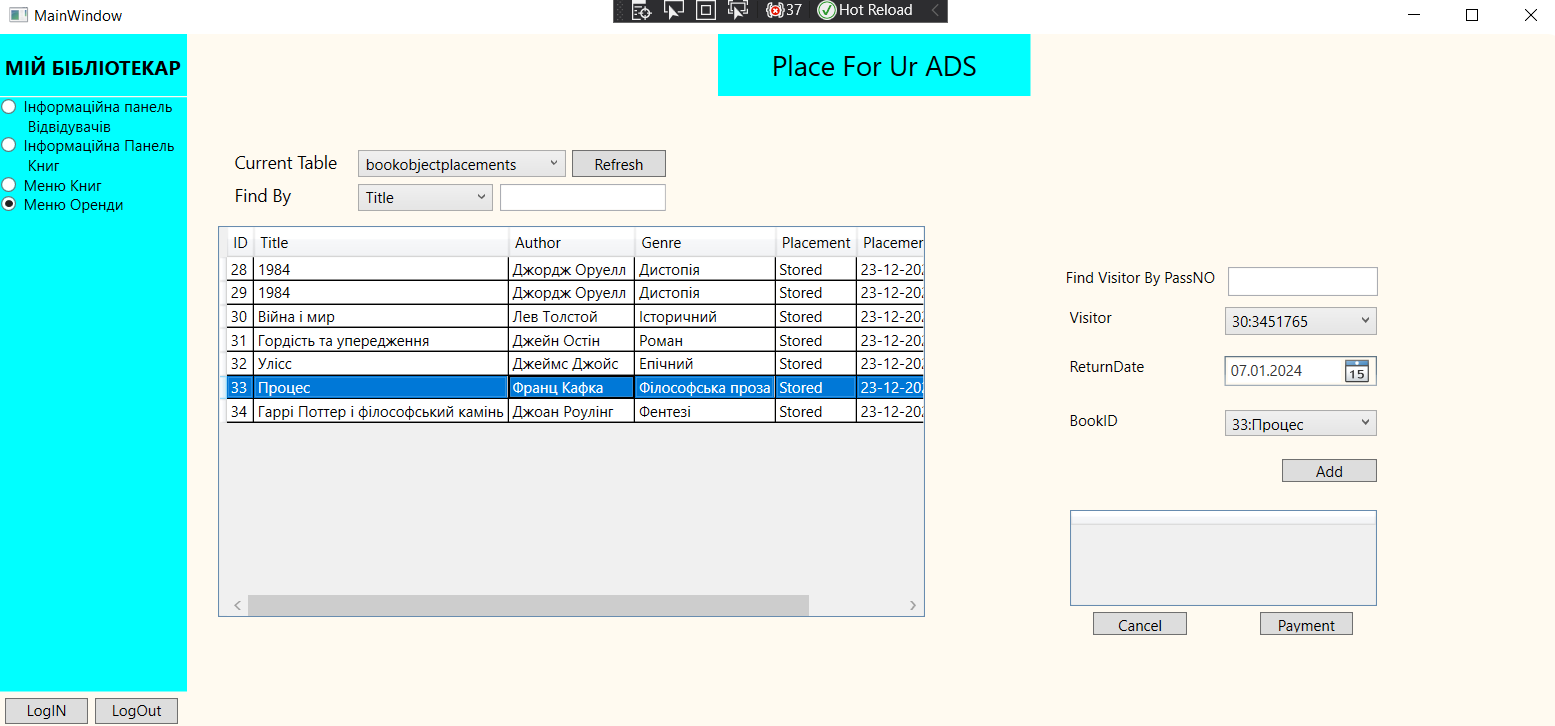


Рисунок 4.7 Створення нового запису до таблиці Оренди

Специфічність цього функціоналу можна описати виконанням додаткових дій. Перш за все це необхідність обору дати повернення та відвідувача.

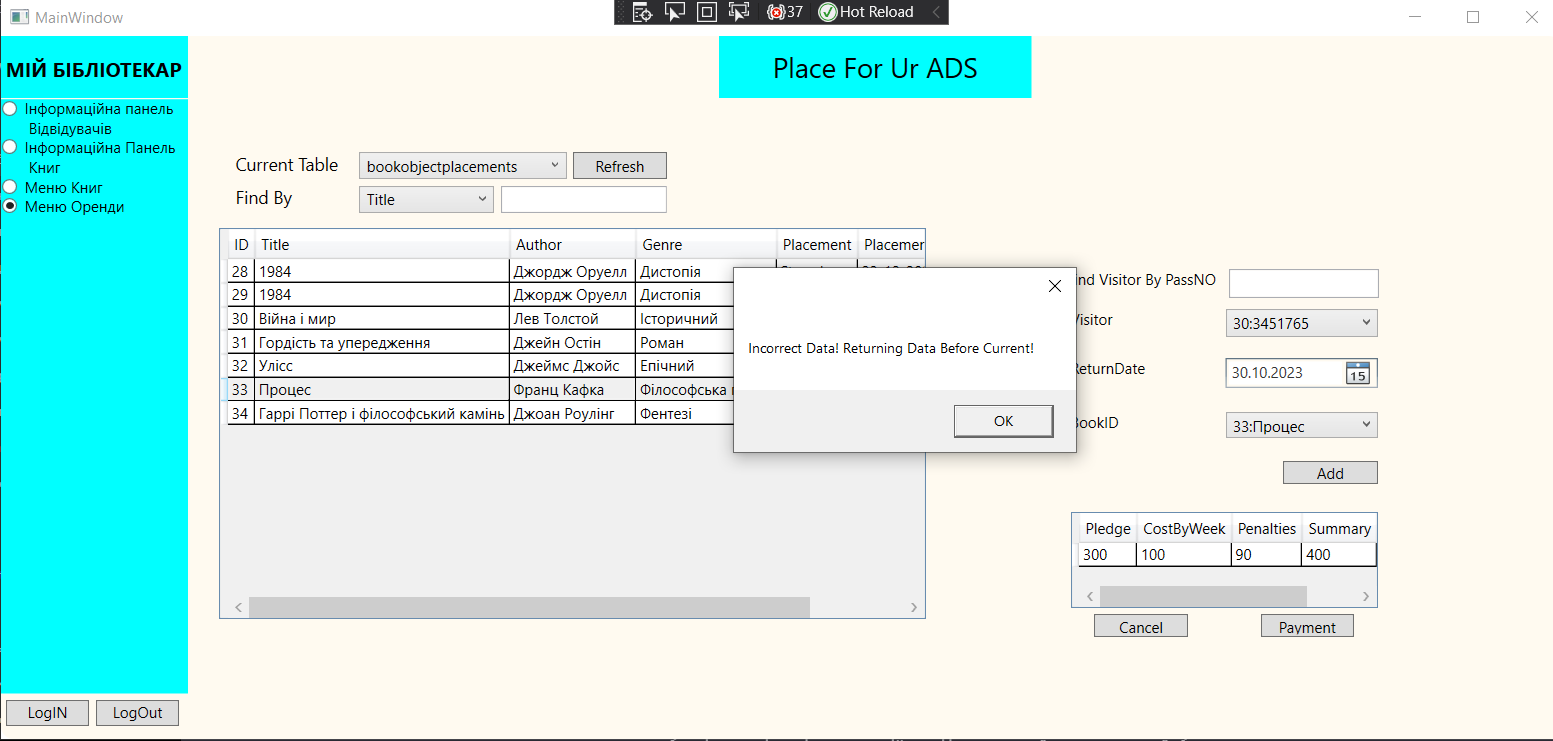


Рисунок 4.8 Помилка при некоректно введених даних

Також варто зазначити що при некоректних даних створення запису неможливе, програма поверне помилку то очистить введені дані.

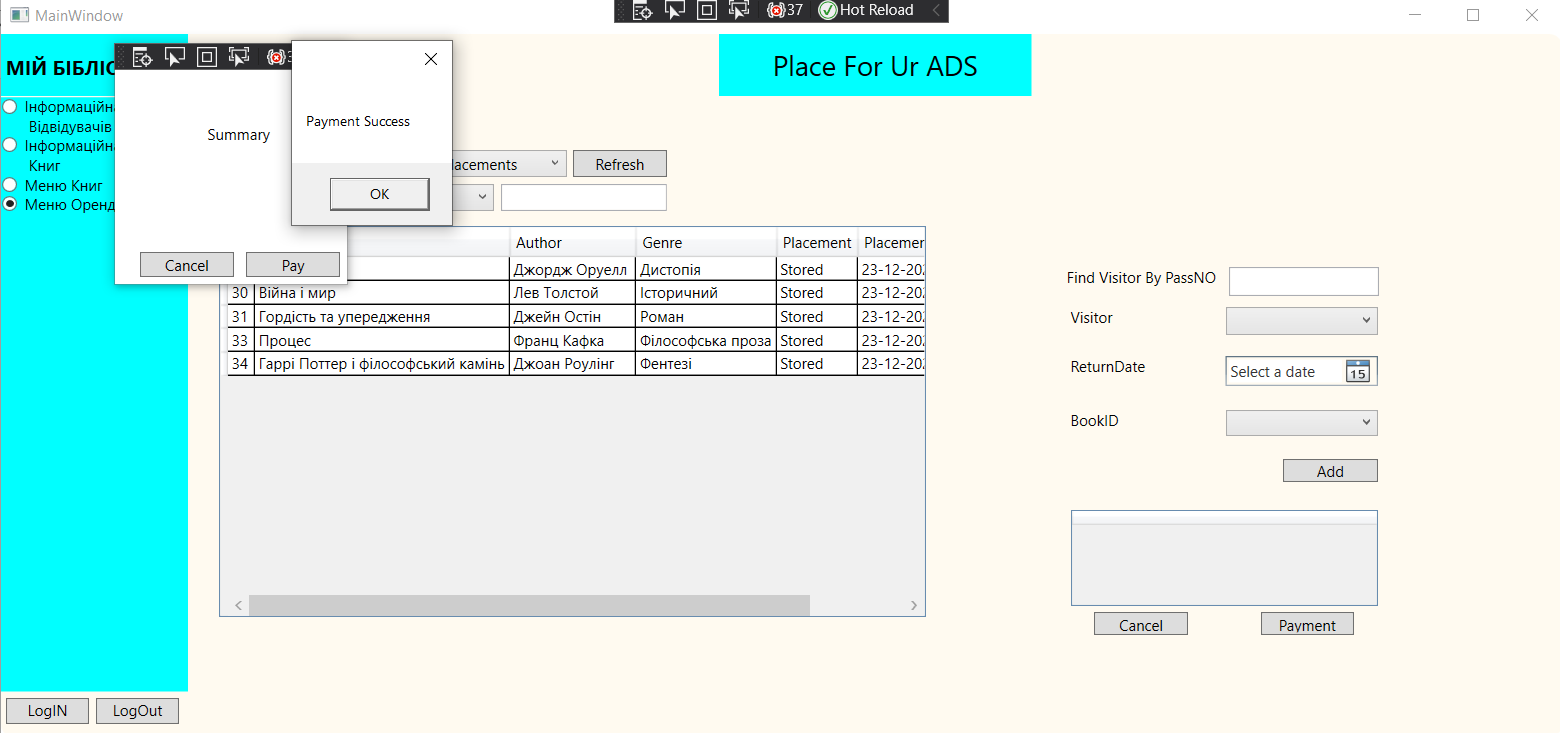


Рисунок 4.9 Приклад Успішного проведення операції

Після Введення коректної інформації користувачу пропонується дві дії:

* Провести оплату – що призведе до імітації способу оплати та створення нового запису Оренди
* Відмінити операцію – що відмінить всі обрані елементи та скине меню обору до початкових значень
  + 1. Редагування існуючих елеменітв

Редагування дених – це важливий елемент взаємодії з базою даних. За рахунок елементів графічного Інтерфейсу можливо змінити значення записів. Бібліотекарю надається повних перелік можливості зміни атрибутів книг. Для цього необхідно обрати необхідний запис та натиснути кнопку «EDIT» це дасть програмі знати, що необхідно провести деякі зміни в вікні додавання елементів, щоб його функціонал можна було задіяти для редагування. Після завершення операції відбувається завантаження усіх відомих змінних обраного запиту (див Рисунок 1.1)

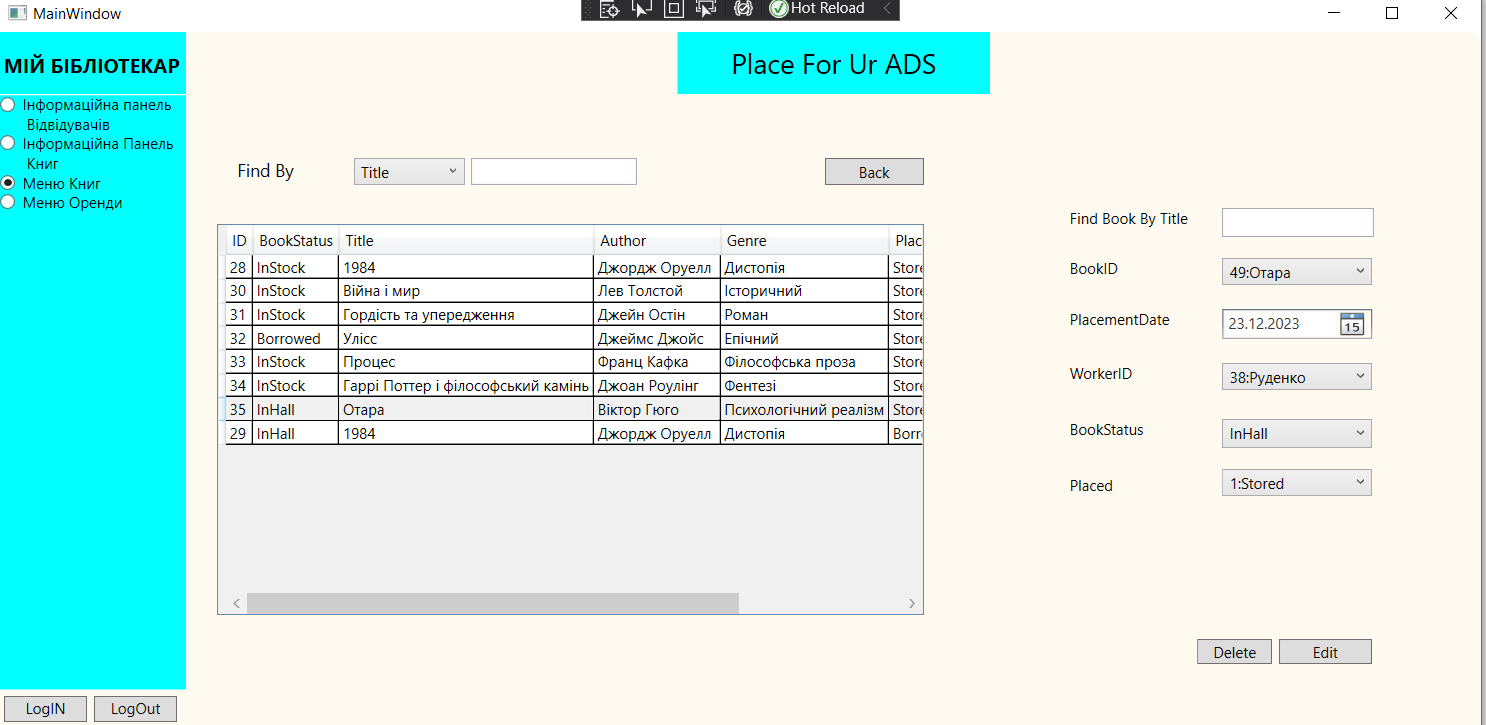


Рисунок 4.10 Взаємодія з меню Редагування в розділі Меню Книг

Ще одним елементом запиту на редагування даних є Зміна статусу Оренди. Його меню має Інтерфейс та кнопки статусу, відповідно до умов повернення книг (Книга повернута ціла та книга пошкоджена) обирається відповідний пункт меню. При пошкоджені книга помічається як пошкоджена і не може бути видана в оренду знов. Також існує механізм, який заборонить змінювати статус Оренди, якщо один з пунктів вже обрано. Роботу цього меню представлено на рисунку 1.1

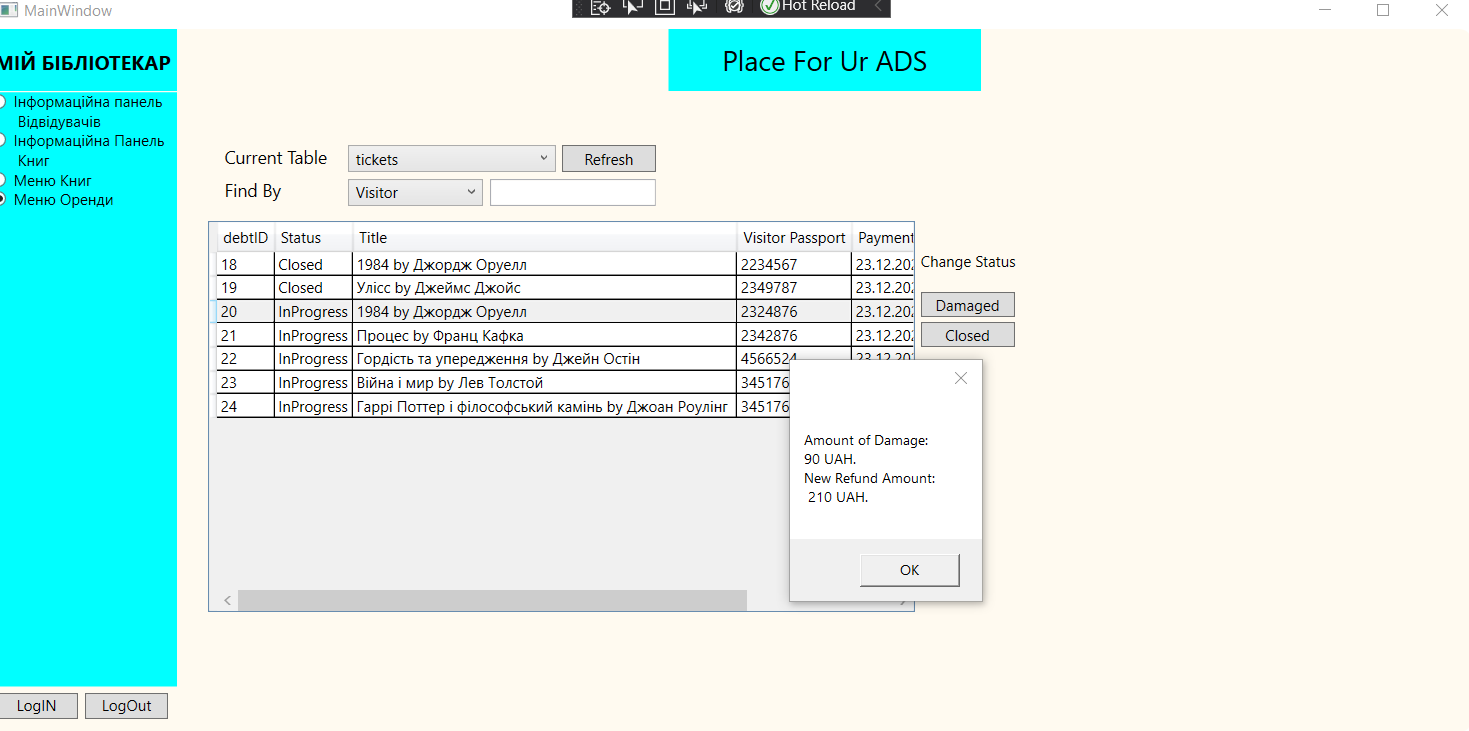


Рисунок 4.11 Приклад Встановлення статусу як «Пошкоджена»

За допомогою редагування запитів можливо збільшити функціонал графічного інтерфейсу і продуктивності додатка для роботи бібліотеки

* + 1. Видалення даних

При помилковому створені запису існує необхідність видалити об’єкт. Функціонал додатка дозволяє користувачеві змінити об’єкт за допомогою видалення. Але важливо мати на увазі можливі наслідки видалення даних. Врахування залежностей між таблицями і спрямованих зв'язків дозволяє уникнути випадкового видалення важливих даних або порушень цілісності бази даних. Загалом, видалення даних - це складний процес, який потребує ретельного планування, обережності та дотримання вимог безпеки

На рисунку 1.1 показана рядок, який має бути видалений.

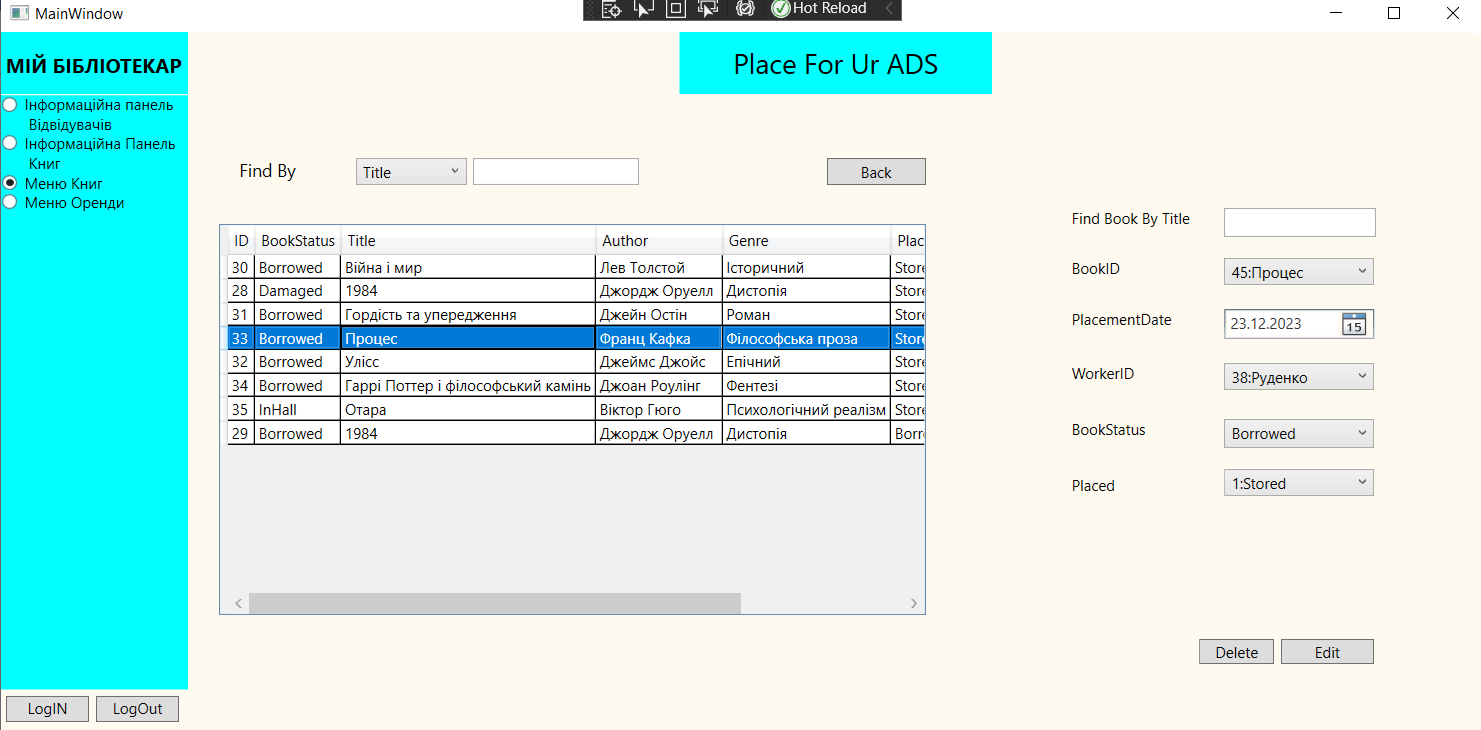
****

Рисунок 4.12 Обраний запис для демонстрації видалення

Після того як обрано рядок та натиснуто на кнопку «Змінити» об’єкт був перенесений до імпровізованого редактора, де з’явилась можливість видалення.

На рисунку 1.2 можна побачити наслідки видалення

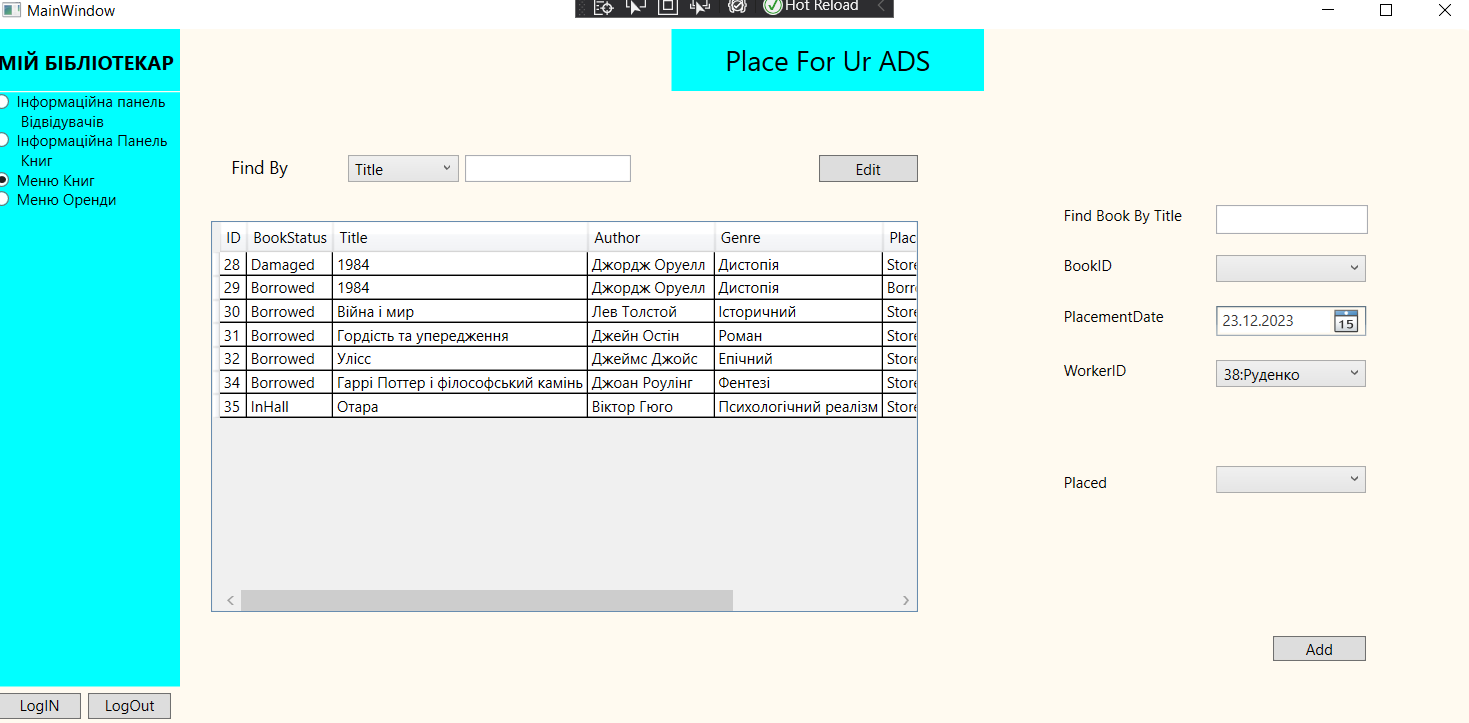


Рисунок 4.13 Видалення обраного запису

Як можна зазначити що в таблиці зник запис з ID 33, який було обрано для видалення, а отже процес пройшов успішно. Таким чином можна взаємодіяти з будь яким об’єктом бази даних, але слід пам’ятати – видалення об’єкту потребує уважності і точності, оскільки випадкове видалення необхідного елементу може призвести до збоїв в роботі бази даних

**ВИСНОВКИ**

Під час виконання Курсової Роботи ознайомлено з відомостями взаємодії та принципу роботи додатку, що створений для керування системою бібліотеки.

В Результаті роботи розроблено програмний продукт у вигляді графічного інтерфейсу баз даних, який здатен керувати роботою Бібліотеки за допомогою запитів до Системи Управління Базами Даних MySQL. А саме створювати редагувати та видаляти записи таблиць.

Інтерфейс програми спроектований за допомогою графічної підсистеми Windows Presentation Foundation та мови програмування C#, що дало змогу ефективно виконувати поставлені задачі.

В Процесі реалізації графічного інтерфейсу використано різні методи для забезпечення цілісності даних та створення надійної системи бази даних. Отримано навички організації бази даних для забезпечення ефективного зберігання та обробки необхідних даних. Запропоновано програмний код, що здатен взаємодіяти з базою даних та допомагати в наданні послуги оренди Літератури.

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. ‘1. Sierra K. Head First SQL. Sebastopol : O’Reilly Media, 2007. 586 p.[Електронний ресурс]-Режим доступу: <https://theswissbay.ch/pdf/Gentoomen%20Library/Databases/O%27Reilly%20Head%20First%20SQL.pdf>
2. Поняття Use Case Diagram [Електронний ресурс][1] – Режим доступу:

<https://circle.visual-paradigm.com/docs/uml-and-sysml/use-case-diagram/>

1. Інтегроване середовище Visual Studio [Електронний ресурс]-Режим доступу:
2. Графічна підсистема WPF [Електронний ресурс]-Режим доступу: <https://learn.microsoft.com/en-us/dotnet/desktop/wpf/overview/?view=netdesktop-8.0>
3. Система Управління Баз Даними MySQL [Електронний ресурс]-Режим доступу: <https://docs.oracle.com/en-us/iaas/mysql-database/doc/creating-db-system1.html#GUID-F2CB52E0-6C6A-407F-A8AC-D82371186D2D>
4. Системи баз даних [Електронний ресурс]-Режим доступу: <https://ela.kpi.ua/bitstream/123456789/27252/1/Seheda.pdf>
5. Концептуальна модель [Електронний ресурс]-Режим доступу: [https://www.visual-paradigm.com/support/documents/vpuserguide/ 3563/3564/ 85378\_conceptual,l.html](https://www.visual-paradigm.com/support/documents/vpuserguide/%203563/3564/%2085378_conceptual,l.html)
6. Посібник по SQL [Електронний ресурс]-Режим доступу: <https://www.w3schools.com/sql/sql_view.asp>
7. Графічна підсистема WPF Посібник [Електронний ресурс] – Режим <https://wpf-tutorial.com/datagrid-control/details-row/>
8. Процедури та Тригери MySQL[Електронний ресурс]-Режим доступу: <https://dev.mysql.com/doc/refman/8.0/en/create-procedure.html>

**ДОДАТОК А**

Додаток для управління Базою Даних Бібліотеки

Лістинг процедур

УКР.НТУУ\_КПІ\_ЦТЕ\_НН ІАТЕ\_ТР-12

Аркушів 3

Київ – 2023

**Процедура обрахунку ціни оренди:**

CREATE DEFINER=`root`@`::1` PROCEDURE `CalculateRentPrice`(

IN depositPrice DECIMAL(10,2),

IN weeklyPrice DECIMAL(10,2),

IN fullWeeks INT,

IN discount DECIMAL(5,2)

)

BEGIN

DECLARE rentPrice DECIMAL(10,2);

IF weeklyPrice = 0 THEN

SET weeklyPrice = 1;

END IF;

SET rentPrice = depositPrice + (weeklyPrice \* fullWeeks) \* discount;

SELECT rentPrice AS 'RentPrice';

END

**Процедура пошуку збіжностей:**

CREATE DEFINER=`root`@`::1` PROCEDURE `SearchSimilarByColumn`(

IN tableName VARCHAR(255),

IN columnName VARCHAR(255),

IN searchText VARCHAR(255)

)

BEGIN

SET @sql\_query = CONCAT('SELECT \* FROM ', tableName, ' WHERE `', columnName, '` LIKE \'%', searchText, '%\'');

PREPARE stmt FROM @sql\_query;

EXECUTE stmt;

DEALLOCATE PREPARE stmt;

END

**Процедура що повератає значення параметрів ціни книги:**

CREATE DEFINER=`root`@`::1` PROCEDURE `GetBooksCostValues`(IN currentBookID INT)

BEGIN

DECLARE valueClassID INT;

DECLARE bookClassID INT;

DECLARE debtsClassID INT;

DECLARE pledgeValue DECIMAL(10,2);

DECLARE costValue DECIMAL(10,2);

DECLARE penaltiesValue DECIMAL(10,2);

Select `BookID` into debtsClassID

from debts

where ID=currentBookID;

Select `BookID` into bookClassID

from bookobjectplacements

where ID=debtsClassID;

SELECT `Value classID` INTO valueClassID

FROM books

WHERE ID = bookClassID;

SELECT Cost, Pledge

INTO costValue, pledgeValue

FROM `library db 3sem`.value\_сlasses

WHERE ID = valueClassID;

SELECT pledgeValue \* `Penalty Value (%)` / 100

INTO penaltiesValue

FROM penalties

WHERE ID = (

SELECT PenaltyID

FROM `library db 3sem`.value\_сlasses

WHERE ID = valueClassID

);

SELECT costValue AS 'Cost', pledgeValue AS 'Pledge', penaltiesValue AS 'Penalties';

END

**Процедура що повертає значення скидки Відвідувача в якості множника:**

CREATE DEFINER=`root`@`::1` PROCEDURE `GetCurrentDiscountMultiplier`(IN currentVisitorID INT)

BEGIN

declare currentDiscount double(5,4);

declare discounttypeID int;

select `library db 3sem`.visitors.`DiscountTypeID`

into discounttypeID

from visitors

where ID = currentVisitorID;

select 1-`library db 3sem`.discounttypes .`DiscountValue(%)`/100

into currentDiscount

from discounttypes

where ID = discounttypeID;

SELECT currentDiscount, discounttypeID , currentVisitorID;

END

**Приклад запиту з створенням запису (функція):**

private void Payment\_Click(object sender, RoutedEventArgs e)

{

if (!currenttransaction.ReadyToPay)

{

return;

}

DateTime time = DateTime.Now;

DateTime InputValue = (DateTime)Date.SelectedDate;

MySqlConnection connection = BD.connection;

String commandText = "SET FOREIGN\_KEY\_CHECKS=0;INSERT INTO `debts` (`ToPayValue`, `VisitorID`, `BookID`, `WorkerID`, `Rent Start Date`, `Return Date`) VALUES (@param1, @param3, @param4, @param5, @param6,@param7)";

connection.Open();

using (MySqlCommand command = new MySqlCommand(commandText, connection))

{

command.Parameters.AddWithValue("@param1", currenttransaction.ToPayValue);

command.Parameters.AddWithValue("@param3", VisitorID.Text.Split(':')[0]);

command.Parameters.AddWithValue("@param4", BookID.Text.Split(':')[0]);

command.Parameters.AddWithValue("@param5", User.getInstance().id);

command.Parameters.AddWithValue("@param6", time);

command.Parameters.AddWithValue("@param7", InputValue);

command.ExecuteNonQuery();

PreLoadFunction();

}

Reset();

}