### НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Кафедра комп’ютерних наук

КУРСОВИЙ ПРОЕКТ

З дисципліни “Організація баз даних”

на тему: “Автоматизоване робоче місце працівника відео прокату”

|  |  |  |  |
| --- | --- | --- | --- |
| Виконала: | **/** | Пилипенко Н.М. ІПЗ-22007Б | **/** |
| Підпис |  | ПІБ, група |  |

### Робота захищена:

**« »** 20 р. Оцінка за національною шкалою: \_ Кількість балів: \_

|  |  |  |
| --- | --- | --- |
| Члени комісії: |  |  |
| (підпис) |  | (прізвище та ініціали) |
| (підпис) |  | (прізвище та ініціали) |

**КИЇВ - 2024**

## ЗМІСТ

1. [Постановка задачі 6](#_bookmark0)
2. [Інформаційне забезпечення 8](#_bookmark1)
   1. [Логічна модель даних у вигляді ER-діаграми 8](#_bookmark2)
   2. [Створення бази даних 10](#_bookmark3)
      1. [Створення таблиць 11](#_bookmark4)
      2. [Створення уявлень 12](#_bookmark5)
      3. [Створення збережених процедур та тригерів 13](#_bookmark6)
      4. [Створення користувачів 14](#_bookmark7)
3. [Прикладне програмне забезпечення 16](#_bookmark8)
   1. [Організаційна структура програмного забезпечення 16](#_bookmark9)
   2. [Вибір інструментарію для створення прикладного програмного](#_bookmark10) [забезпечення 16](#_bookmark10)
   3. [Алгоритмізація та програмування програмних модулів 17](#_bookmark11)
      1. [Реалізація інтерфейсу користувача 17](#_bookmark12)
      2. [Забезпечення інтерфейсу з базою даних 18](#_bookmark13)
      3. [Формування звітної інформації 19](#_bookmark14)
4. [Рекомендації щодо впровадження та експлуатації системи 20](#_bookmark15)
   1. [Тестування системи 20](#_bookmark16)
   2. [Вимоги до апаратного та програмного забезпечення 2](#_bookmark17)6
      1. [Вимоги до апаратного забезпечення 2](#_bookmark18)6
      2. [Вимоги до програмного забезпечення 2](#_bookmark19)6

### СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

|  |  |  |
| --- | --- | --- |
| ВП | – | Відео-прокату |
| ПК | – | персональний комп’ютер |
| БД | – | база даних |
| ПЗ | - | Програмне забезпечення |

**ВСТУП**

Основні ідеї сучасних інформаційних технологій, в основному, базуються на концепції, згідно якої усі дані повинні бути організовані у, так би мовити, сховищах даних (базах даних) з метою систематизації інформації та задоволення інформаційних потреб користувачів.

Розробка і реалізація такого програмного продукту, яке буде зберігати у собі дані ВП, є досить актуальним, адже це зменшить кількість роботи з паперами, дозволить затрачати менше часу для пошуку потрібної інформації та пришвидшить виконання завдань, пов’язані з даними та обслуговуванням. ПЗ дозволяє пришвидшити процес обміну інформацією шляхом їх цифровізації та наступної передачі через локальну мережу та/або електронну пошту. Відповідно, це зменшить затрати часу на обмін інформацією у ВП.

**Актуальністю** теми курсового проекту полягає в тому, що з переходом до зберігання даних у базах даних потрібна нова організація цих самих даних. Це дає суттєву необхідність замінити застарілу, мало ефективну систему збереження інформації, яка значно затримувала не лише передачу даних у ВП, але і самий його розвиток.

**Метою роботи та завданням дослідження** є розробка автоматизованого робочого місця працівника ВП.

**Об’єктом дослідження** є інформація ВП, яка буде оброблятися програмним продуктом.

**Предметом дослідження** є створення програми для обліку та впорядкування даних про відео продукцію, яка закріплена за певним покупцем.

**Методи** розробки базуються на технологіях мови програмування С#, Microsoft SQL Server із використанням програмного середовища розробки Microsoft Visual Studio та середовища створення та адміністрування БД.

**Практичне значення одержаних результатів** полягають в розробці програмного продукту для обліку відео матеріалів, за які несе відповідальність покупець який здійснює прокат.

Для досягнення мети курсової роботи поставлено такі завдання:

* Розглянути основні принципи організації даних відео прокату в сучасних умовах, структуру і особливості існуючих форм і систем збереження даних;
* Дати характеристику та описати предметну область розроблюваного програмного продукту;
* На основі даних аналізу, розробити практичні рекомендації, щодо оптимізації вдосконалення організації інформації для продавця ВП.

# 1 ПОСТАНОВКА ЗАДАЧІ

Виходячи із специфіки діяльності продавця перш за все необхідно забезпечити доступ до інформації щодо відео які знаходяться в прокаті, їх жанрів та клієнтів. З цього випливає, що у проект потрібно включити інформацію про: жанри, відео та клієнтів які здійснюють прокати та звісно ж і самі прокати.

Основними методами обробки інформації програмним продуктом є:

* Пошук даних;
* Видалення даних;
* Додання даних ;
* Редагування даних;
* Оформлення звітної інформації;
* Обробка звітної документації іншими ПЗ.

Ввід даних у програму буде відбуватися користувачем через візуальні компоненти програмного продукту.

Основні ***вхідні*** дані, які будуть представлені у програмному продукту – прокати, а саме інформація про відео яке взяте та ким тобто данні про покупця і на який термін.

Основними ***вихідними*** даними буде відео які знаходяться в данному відео прокаті та прокати які здійснювалися ця інформація буде виводитися з БД на екран комп’ютера та у формі електронного чи паперового звіту.

**Результатом роботи програми** є взаємодія із базою даних та її складовими (збереженими процедурами, які частково і визначають функціонал розроблюваного

програмного забезпечення, таблицями, уявленнями та тригерами) та звітною інформацією.

Приклад звітної інформації буде представлено у **Додатку А.**

Насамперед, для успішного створення програмного продукту, потрібно вирішити ряд таких задач:

* Побудувати первісний набір відношень (сутностей);
* Для кожного атрибуту відношень визначити тип даних;
* Виконати нормалізацію первісного набору (мінімум до третьої нормальної форми);
* Створити та виконати запити для створення відношень та інших об’єктів БД;
* Реалізувати фізичний проект – інтерфейс користувача;
* Створити уявлення, процедури, транзакції та тригери, які будуть необхідні для управління інформацією у БД;
* Надати різні права доступу для користувача і адміністратора.
* Скомпонувати інтерфейс користувача та реалізацію БД.

# 2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

В проекті будуть наступні сутності:

* Жанри;
* Відео;
* Покупці;
* Прокати;

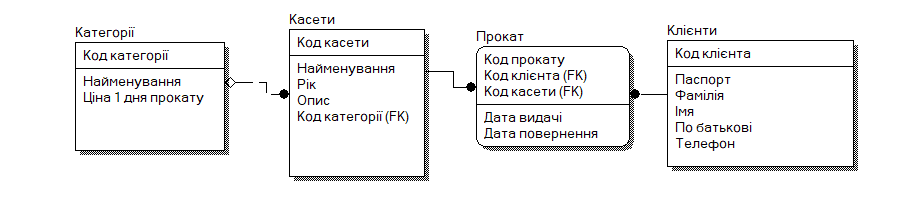
Детальніше про використовувані типи даних у сутностях див. пункт 2.2.1. та

**Додаток Б.**

## Логічна модель даних у вигляді ER-діаграми

Нормальна форма – властивість відношення в реляційній моделі даних, яка характеризує його з точки зору надмірності, яка потенційно може призвести до логічно помилкових результатів.

Логічна модель до нормалізації представлена таким чином:



#### Перша нормальна форма.

**Рис. 2.1**. ER - модель

Для доведення, що модель знаходиться у 3 НФ необхідно довести, що вона також знаходиться у 1НФ та 2НФ.

Відношення знаходиться у 1НФ, якщо всі його атрибути є простими, всі використовувані домени повинні містити у собі лише скалярні значення. Не повинно бути також повторень рядків таблиці.

#### Друга нормальна форма.

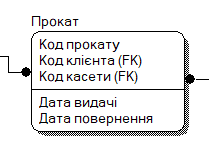
Відношення знаходиться у 2НФ, якщо воно знаходиться у 1НФ і кожен неключовий атрибут залежить від первинного ключа.

Зв’язки між сутностями у моделі змінені на неідентифікуючі задля виконання усіх вимог другої нормальної форми.

#### Третя нормальна форма.

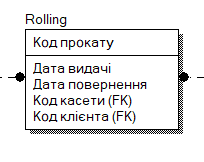
Відношення знаходиться у 3НФ, якщо воно знаходиться у 2НФ і кожен неключовий атрибут нетранзитивно залежить від первинного ключа.

На рис. 2.1 видно, що модель не є у 3НФ, оскільки ключі в сутності «Прокат» є неправильно визначеними.



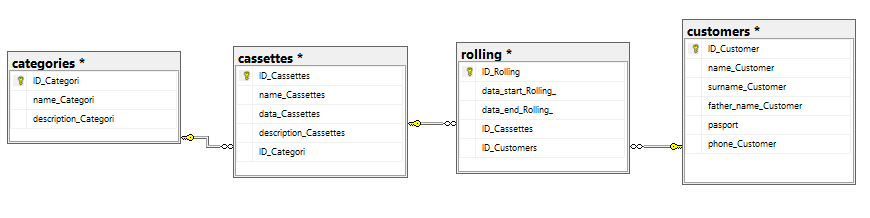
**Рис. 2.2.** Сутність «Прокат»

Пісня внесення змін і зведення до 3НФ.



**Рис. 2.3.** Сутність «Прокат»

Після проведення нормалізації логічна модель у вигляді ER – діаграми автоматизованого робочого місця продавця ВП у 3 нормальній формі буде виглядати ось так:



**Рис. 2.4**. Діаграма створеної бази даних

## Створення бази даних

Мова SQL використовується не лише для обробки інформації, але й призначена для виконання усіх операцій з базами даних і таблицями, включаючи також створення таблиць та робота з ними. Існує два способи створення таблиць:

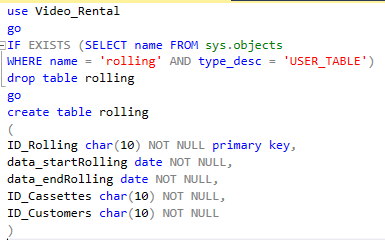
* + 1. більшість СУБД володіють візуальним інтерфейсом для інтерактивного створення таблиць та управління ними;
    2. таблицями також можна маніпулювати, використовуючи оператори SQL. Варто зазначити, що, коли ви використовуєте інтерактивний інструментарій СУБД, насправді уся робота виконується операторами SQL.

Для створення таблиць програмним способом використовують оператор **CREATE DATABASE**. Для цього потрібно вказати ім'я таблиці, яке вказується після ключового слова **CREATE DATABASE**;



**Рис. 2.5.** SQL-запит на створення БД

* + 1. **Створення таблиць.** Розглянемо детально на **Рис. 2.6** SQL запит на створення таблиці «Прокат»:



**Рис. 2.6** SQL запит на створення таблиці

Для створення таблиць програмним способом використовують оператор **CREATE TABLE**. Для цього потрібно вказати наступні дані:

* ім'я таблиці, яке вказується після ключового слова **CREATE TABLE**;
* імена та визначення стовпців таблиці, що відділені комами;
* в деяких СУБД також вимагається, щоби було вказано місце розташування таблиці.

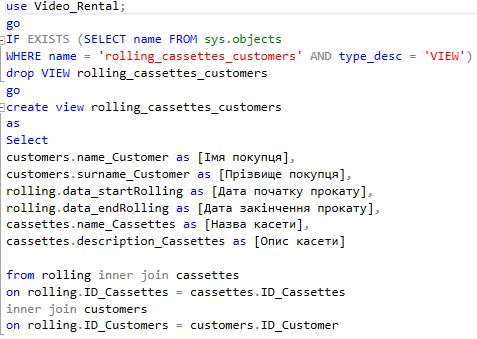
Так ми спочатку вказуємо назву нової таблиці, потім в дужках перераховуємо стовпці, які будемо створювати, причому їх назви не можуть повторюватися в межах однієї таблиці. Після назв стовпців вказується тип даних для кожного поля, потім зазначаємо чи може поле містити порожні значення (**NULL** або **NOT NULL**), а також потрібно вказати поле, яке буде первинним ключем (**Primary key**).

Мова SQL також дозволяє визначати для кожного поля значення по замовчуванню, тобто, якщо користувач не вкаже значення для певного поля - воно буде автоматично проставлене СУБД. Значення по замовчуванню визначається

ключовим словом **DEFAULT** при визначенні стовпців оператором **CREATE TABLE**.

* + 1. **Створення уявлень.** Для продавця найголовніше мати змогу бачити усю детальну інформацію щодо прокатів (дані про відео, покупця та час прокату). Для цього найкращим варіантом буде створити уявлення для представлення цих даних.

Уявлення, яке відображає повну інформацію про прокати зображено нижче на **Рис. 2.7**:



**Рис. 2.7** SQL запит на створення уявлення

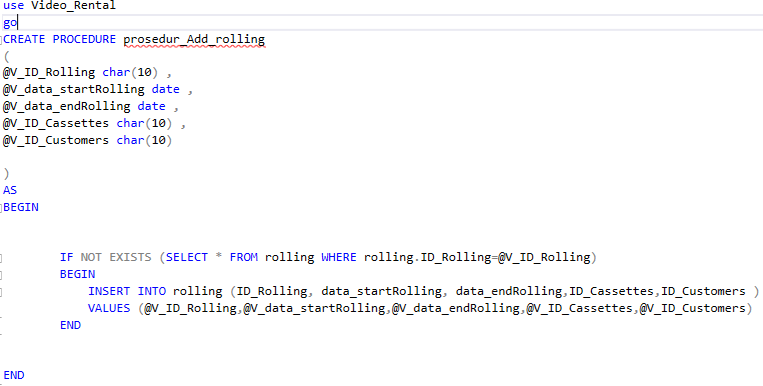
Для роботи з базою даних було створено такі уявлення:

* **rolling\_cassettes\_customers** – перегляд інформації про здійснені прокати;
* **cassettes\_categories** – перегляд інформації щодо відео та їх жанрів які є в наявності;

Детальніше переглянути усі уявлення можна у **Додатку В**.

* + 1. **Створення збережених процедур та тригерів.** Для коректної роботи програмного продукту необхідно було розробити наступні процедури:
* Додавання даних у сутності (жанри, відео, покупці, прокати);
* Редагування у сутності (жанри, відео, покупці, прокати);;
* Видалення у сутності (жанри, відео, покупці, прокати);;

Розглянемо приклад процедури для додавання даних про прокат на **Рис. 2.8**:



**Рис. 2.8** SQL запит на створення процедури

У наведеній процедурі задаються вхідні параметри, які будуть оброблені програмно. В результаті чого користувач зможе взаємодіяти із атрибутами БД та виконувати певні дії.

Інші програмні процедури будуть представлені у **Додатку Г**.

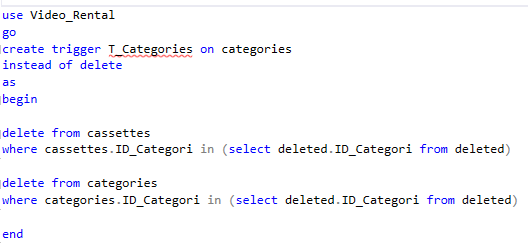
#### Тригери БД

Для коректної роботи програми було створені такі тригери:

* Тригер на видалення даних з сутності «Жанри»;
* Тригер на видалення з сутності «Покупці»;

Приклад тригера для сутності Жанри представлений на **Рис**

**2.9** наступним чином:



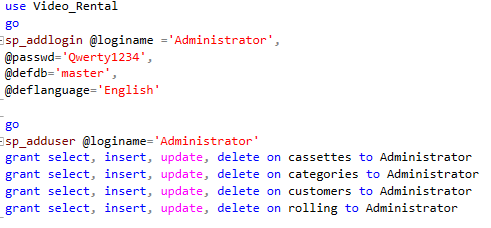
**Рис. 2.9** SQL запит на створення тригера

У даному прикладі тригер виконується при спробі видалення непопулярного жанру який не буде представлений у даному ВП.

Інші тригери будуть представлені у **Додатку Г**.

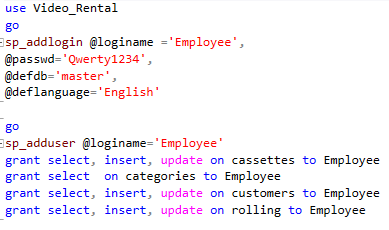
* + 1. **Створення користувачів.** Створення користувачів та ролей необхідно для програмного продукту для розділення прав та повноважень щодо керуванням даними за допомогою ПЗ. Для цього було створено дві основні ролі: адміністратор і продавець. Адміністратор має усі унікальні права для роботи з БД та її даними. Користувач має лише обмежені права, а саме: дії з таблицею Жанри йому обмежені лише до перегляду і він не має права видалення будь яких записів

Створення адміністратора представлено на **Рис. 2.10**:



**Рис. 2.10** SQL запит на створення адміністратора

Створення продавця представлено на **Рис. 2.11**:

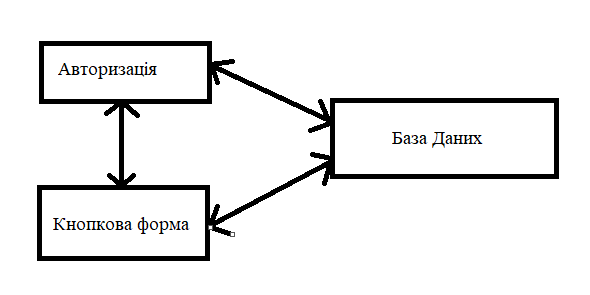


**Рис. 2.11** SQL запит на створення продавця

# 3 ПРИКЛАДНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

## Організаційна структура програмного забезпечення

Блок – схема організаційної структури програмного забезпечення виглядає таким чином:



**Рис. 3.1.** Блок – схема проекту

У програмі для працівника ВП були передбачені наступні вікна (форми) для роботи:

* Форма авторизації;
* Компоненти DataModule, які містять екземпляри для роботи зв’язку з базою даних;
* Користувацька форма, яка використовується для перегляду та взаємодією із даними.

## Вибір інструментарію для створення прикладного програмного забезпечення

Для створення автоматизованого робочого місця працівника ВП було використане середовище програмування Microsoft Visual Studio 2019 та мова програмування C#.

Microsoft Visual Studio — серія продуктів фірми Майкрософт, які включають інтегроване середовище розробки програмного забезпечення та низку інших інструментальних засобів. Ці продукти дозволяють розробляти як консольні програми, так і програми з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-застосунки, веб-служби як в рідному, так і в керованому кодах для всіх платформ, що підтримуються 13 Microsoft Windows, Windows Mobile, Windows Phone, Windows CE, .NET Framework, .NET Compact Framework та Microsoft Silverlight.Connection (представляє підключення до віддаленого джерела даних):

* Query (виконує SQL-запити та представляє набір рядків, отриманий від джерела даних)
* Command (використовується для виконання команд і SQL-запитів з параметрами) –
* DataSource (може представляти одну запис об'єкта Recordset або ж ієрархічну структуру, що складається з текстових даних)

Для створення звітів використовується RDLC Report - сучасне рішення для програмного забезпечення будь-якого рівня.

Переваги даного рішення:

* Повноцінний сучасний редактор, інструментарій якого містить велику кількість засобів для візуального оформлення, створення, налаштування та змінювання шаблонів звітів.
* Простий та зрозумілий класичний інтерфейс.
* Багатосторінкові та багато шаблонні звіти

## Алгоритмізація та програмування програмних модулів

* + 1. **Реалізація інтерфейсу користувача.** Для реалізації інтерфейсу користувача було використано стандартні компоненти та об’єкти Visual Studio, а саме:
       - Windows Form;
       - Label;
       - TextBox;
       - DataGridView;
       - Button; - TabControl;
       - comboBox; - MessageBox;

Детальніше про реалізацію програмного інтерфейсу можна ознайомитися у розділі «Тестування системи».

* + 1. **Забезпечення інтерфейсу з базою даних.** Основою інтерфейсу з базою даних є об’єкт DBConnection. Саме він встановлює з’єднання з базою даних. Основною властивістю DBConnection є ConnectionString.

Вікно авторизації призначене для конструбвання параметру Connection компоненту DBConnection, шаблоном якого служить створений файл із рядком підключення виду: ***Provider=SQLOLEDB.1;Persist Security Info=False;Initial Catalog=Inventory\_system;Data Source=DESKTOP-1RPP5E0\SQLEXPRESS.***

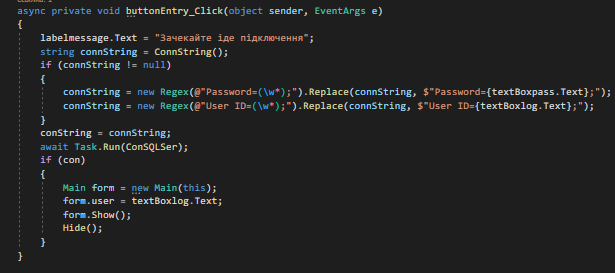
Слід зазначити, що параметри постачальника послуг підключення та базу даних визначено заздалегідь у Visual Studio.

Код для встановлення зчитування рядку підлючення з файлу представлено на **Рис. 3.2**:



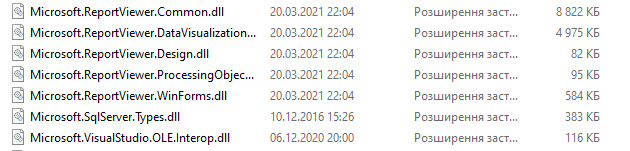
**Рис. 3.2** Код для зчитування рядка підлючення з файлу

Код для авторизації у програмі представлено на **Рис. 3.3**:



**Рис. 3.3** Код для перевірки авторизації

* + 1. **Формування звітної інформації.** Створення звітів відбувається за допомогою бібліотек, які були підключені до середовища програмування:



**Рис. 3.4** Файли, які необхідні для створення звіту

Звіт розроблювався у Visual Studio у конструкторі звітів.

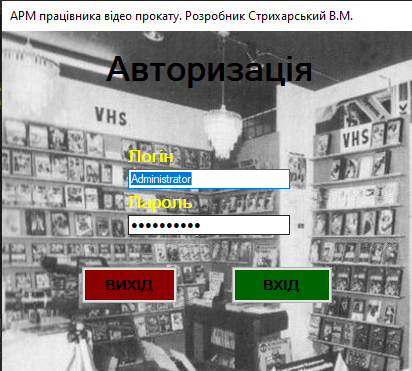
Інші звітна інформація представлена у **Додатку А.**

# 4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ

## Тестування системи

Важливою складовою є тестування програмного забезпечення перед його подальшим впровадженням. Оскільки спеціалізовані функціональні тести (модульні, навантажувальні, тощо) не входять до основної мети даного курсового проекту, в будь – якому разі є необхідним перевірити функціонал створеного додатку. Представимо варіанти роботи системи у вигляді вікон програми, тим самим освітлимо дослідну експлуатацію.

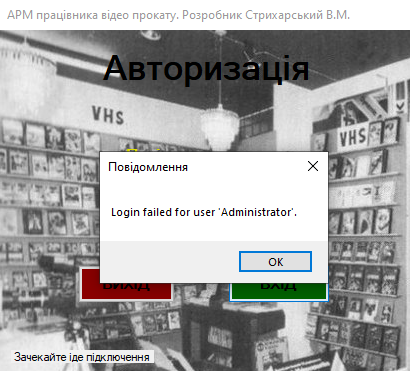
У програмному продукті сторінка авторизації представлена таким чином:



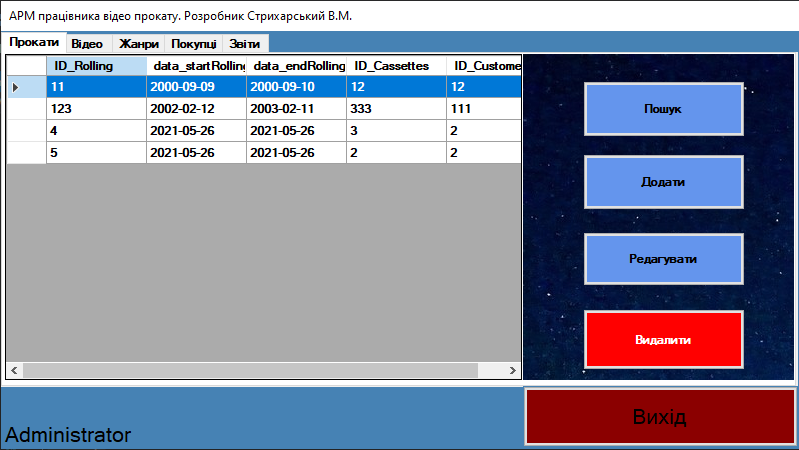
**Рис.4.1** Форма авторизації

Користувач вписує логін та пароль та авторизується у системі.

Компоненти, які були використані у даній формі – Label, TextBox, Button. Якщо користувача немає у системі – виводиться повідомлення про помилку входу:



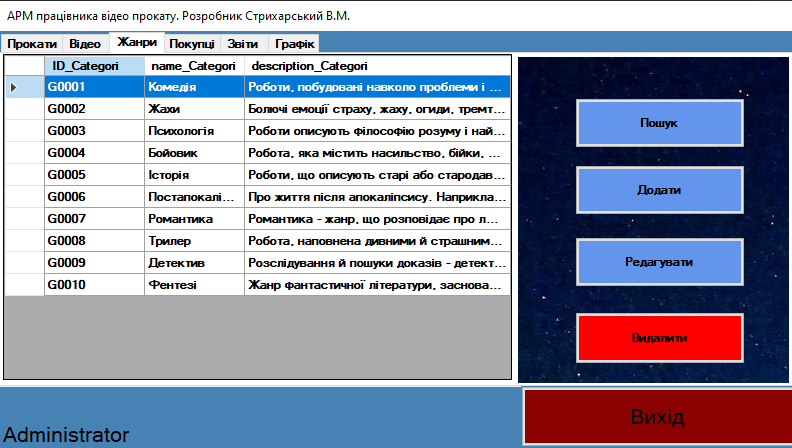
**Рис.4.2** Форма авторизації



**Рис. 4.3** Головне меню програми.

Далі перед користувачем постає інтерфейс головного меню. Кнопки «Скрити» та

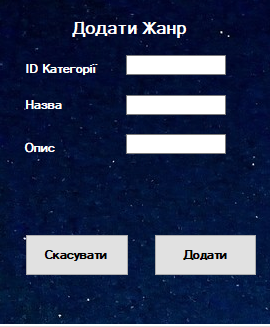
«Закрити» були відключені. Робоча область активується при натисненні на будь – якій з вкладок.



**Рис. 4.4** Головне меню програми

**Кнопка №1.** Додати.

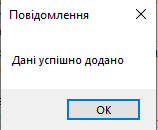
На робочій області з лівого боку розташований компонент DataGridView, який відображає усі додані жанри. З правого боку користувач може додавати дані про них в систему, а саме вписувати такі дані: Код жанру, назва жанру, опис жанру. Після натискання кнопки «Додати» у DataGridView автоматично буде відображена додана інформація.



**Рис. 4.5** Додавання нового жанру

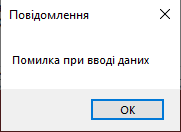
Компоненти, які були використані – Button, TextBox, Label, Panel, DataGridView.

При натисненні на кнопку «Додати» з'являється повідомлення про успішне додавання матеріальної цінності.



**Рис. 4.6** Повідомлення про успішну операцію

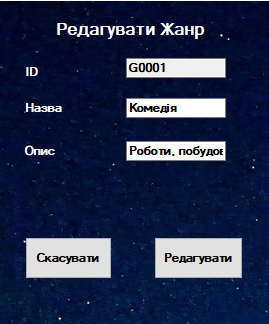
Якщо користувач захоче додати у систему новий жанр з ідентичним «ID» – виведеться помилка.



**Рис. 4.7** Повідомлення про невдалу операцію

**Кнопка №1.** Редагувати.

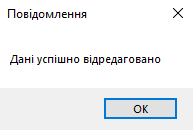
З правого боку користувач може редагувати дані про них в систему, а саме вписувати такі дані: Код жанру, назва жанру, опис жанру ці дані автоматично підлягуються з DataGridView. Після натискання кнопки «Редагувати» у DataGridView автоматично буде відображена відредагована інформація.



**Рис. 4.5** Редагування жанру

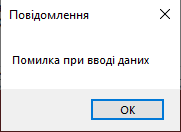
Компоненти, які були використані – Button, TextBox, Label, Panel, DataGridView.

При натисненні на кнопку «Редагувати» з'являється повідомлення про успішне редагування матеріальної цінності.



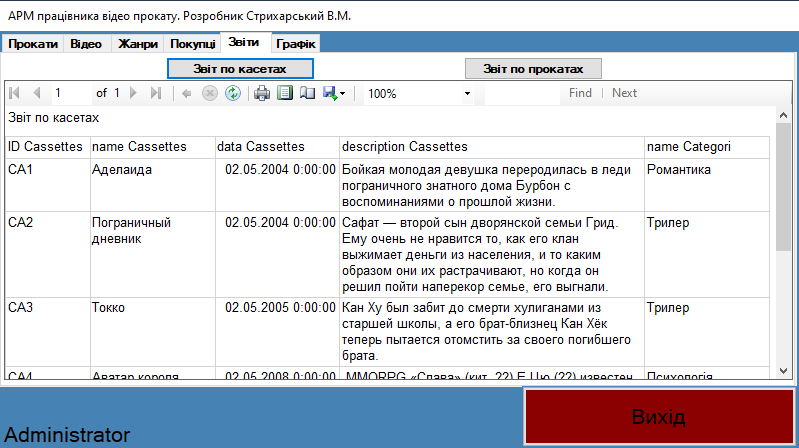
**Рис. 4.6** Повідомлення про успішну операцію

Якщо користувач захоче вказати у систему жанр з ідентичним «ID» – виведеться помилка.



**Рис. 4.7** Повідомлення про невдалу операцію

У вкладці звіти користувач може сформувати звіт за прокатами та відео.



**Рис. 4.8** Звіт сутністю «Відео»

Користувач може імпортувати звіт або відразу ж роздрукувати. Приклад звітної інформації можна побачити у **Додатку А.**

## Вимоги до апаратного та програмного забезпечення

* + 1. **Вимоги до апаратного забезпечення.** Тестування показали наступні мінімальні характеристики для користування ПЗ: ОС Windows 10, оперативна пам’ять об’ємом не менше 500 Мб, процесор з тактовою частотою не менше 133 MHz, наявність монітора, клавіатури, мишки.
    2. **Вимоги до програмного забезпечення. *Вимоги до системного програмного забезпечення.*** Вимоги до системного ПЗ:
       - ОС – Windows 7 та новіші версії;
       - Microsoft.Net Framework 2.0.

## ВИСНОВКИ

Метою курсової роботи було вивчити та критично оцінити ситуацію досліджуваної предметної області, методику, методологію збереження необхідної інформації для працівника ВП та розробити пропозиції щодо удосконалення роботи з обробкою даних, а саме автоматизоване робоче місце працівника відео прокату.

На етапі планування мною було створено аналіз вимог майбутнього програмного продукту та детальний опис функцій, які входять до процесу роботи відділу.

Також було створено базу даних, розроблено уявлення, таблиці, процедури, тригери.

Результатом курсової роботи є пояснювальна записка та розроблений програмний продукт.

При реалізація програмного продукту більшість часу було витрачено на розробку гнучкого функціоналу і розробку максимально простого інтерфейсу , який був би легко адаптивним до персональних комп’ютерів, які знаходяться у вжитку у відео прокатах.

Основна:

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гектор Гарсиа-Молина. Системы баз данных. Полный курс. / Гектор Гарсиа-Молина , Джеффри Д. Ульман, Дженнифер Видом. – М., Спб., К.:

«Издательский дом ВИЛЬЯМС» , 2003. – 1088 с.

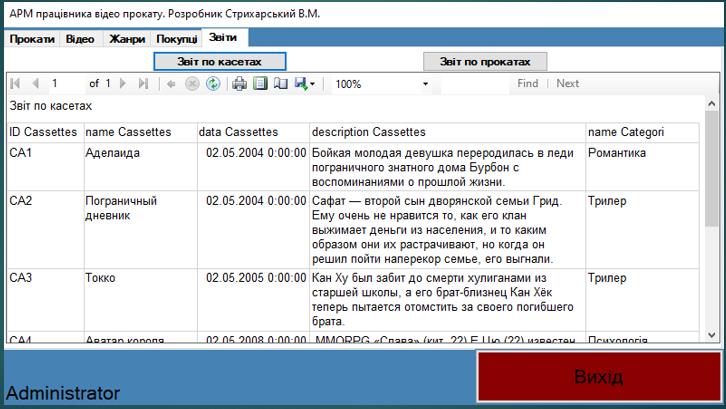
1. К.Дж.Дейт. Введение в системы базы данных. / К.Дж.Дейт. – М., Спб., К.:

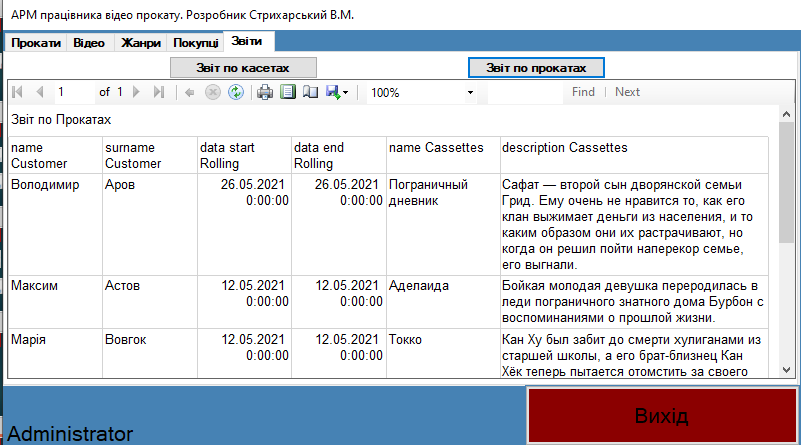
«Издательский дом ВИЛЬЯМС» , 2005. – 1325 с.

1. Пасічник В. В. Організація баз даних і знань. / Пасічник В. В., Резніченко В. А. – К.: BHV, 2006. – 384 с.

### ДОДАТОК А

ПРИКЛАД ЗВІТНЬОЇ ІНФОРМАЦІЇ





### ДОДАТОК Б

SQL-ЗАПИТИ ДЛЯ СТВОРЕННЯ ТАБЛИЦЬ ТА ЗВ’ЯЗКІВ МІЖ НИМИ

use Video\_Rental

go

IF EXISTS (SELECT name FROM sys.objects

WHERE name = 'rolling' AND type\_desc = 'USER\_TABLE')

drop table rolling

go

IF EXISTS (SELECT name FROM sys.objects

WHERE name = 'cassettes' AND type\_desc = 'USER\_TABLE')

drop table cassettes

go

IF EXISTS (SELECT name FROM sys.objects

WHERE name = 'categories' AND type\_desc = 'USER\_TABLE')

drop table categories

go

IF EXISTS (SELECT name FROM sys.objects

WHERE name = 'customers' AND type\_desc = 'USER\_TABLE')

drop table customers

go

create table categories

(

ID\_Categori char(10) NOT NULL primary key,

name\_Categori varchar(20) NOT NULL,

description\_Categori varchar(400) NOT NULL,

)

go

create table customers

(

ID\_Customer char(10) NOT NULL primary key,

name\_Customer varchar(40) NOT NULL,

surname\_Customer varchar(40) NOT NULL,

father\_name\_Customer varchar(40) NOT NULL,

pasport varchar(30) NOT NULL,

phone\_Customer varchar(12) NOT NULL

)

go

create table cassettes

(

ID\_Cassettes char(10) NOT NULL primary key,

name\_Cassettes varchar(20) NOT NULL,

data\_Cassettes date NOT NULL,

description\_Cassettes varchar(400) NOT NULL,

ID\_Categori char(10) NOT NULL

)

go

create table rolling

(

ID\_Rolling char(10) NOT NULL primary key,

data\_startRolling date NOT NULL,

data\_endRolling date NOT NULL,

ID\_Cassettes char(10) NOT NULL,

ID\_Customers char(10) NOT NULL

)

### ДОДАТОК В

SQL-ЗАПИТИ ДЛЯ СТВОРЕННЯ УЯВЛЕНЬ

Сторінка – 1

***Касети***

use Video\_Rental

;

go

IF EXISTS (SELECT name FROM sys.objects

WHERE name = 'cassettes\_categories' AND type\_desc = 'VIEW')

drop VIEW cassettes\_categories

go

create view cassettes\_categories

as

Select

cassettes.name\_Cassettes as [Назва касети],

cassettes.data\_Cassettes as [Дата випуску],

cassettes.description\_Cassettes as [Опис касети]

from cassettes inner join categories

on categories.ID\_Categori = cassettes.ID\_Categori

…………………………………………………………………………………………

#### Прокати

use Video\_Rental

;

go

IF EXISTS (SELECT name FROM sys.objects

WHERE name = 'rolling\_cassettes\_customers' AND type\_desc = 'VIEW')

drop VIEW rolling\_cassettes\_customers

go

create view rolling\_cassettes\_customers

as

Select

customers.name\_Customer as [Імя покупця],

customers.surname\_Customer as [Прізвище покупця],

rolling.data\_startRolling as [Дата початку прокату],

rolling.data\_endRolling as [Дата закінчення прокату],

cassettes.name\_Cassettes as [Назва касети],

cassettes.description\_Cassettes as [Опис касети]

from rolling inner join cassettes

on rolling.ID\_Cassettes = cassettes.ID\_Cassettes

inner join customers

on rolling.ID\_Customers = customers.ID\_Customer

### ДОДАТОК Г

SQL-ЗАПИТИ ДЛЯ СТВОРЕННЯ ПРОЦЕДУР ТА ТРИГЕРІВ

Сторінка – 2

#### Процедура для Додавання даних у сутності Жанри

#### 

#### Процедура для Редагування даних у сутності Жанри

#### 

#### Процедура для Видалення даних у сутності Жанри

#### 

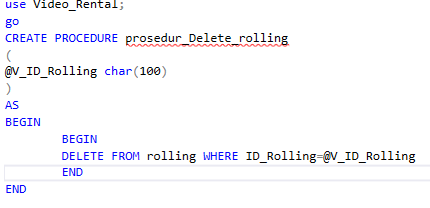
#### Процедура для Додавання даних у сутності Прокат

#### 

#### Процедура для Редагування даних у сутності Прокат

#### 

#### Процедура для Видалення даних у сутності Прокат



#### Процедура для Додавання даних у сутності Відео

#### 

#### Процедура для Редагування даних у сутності Відео

#### 

#### Процедура для Видалення даних у сутності Відео

#### 

#### Процедура для Додавання даних у сутності Покупець

#### 

#### Процедура для Редагування даних у сутності Покупець

#### 

#### Процедура для Видалення даних у сутності Покупець

#### 

#### Тригер для видалення Жанру

#### 

#### Тригер для видалення Покупця

#### 

#### ДОДАТОК Д

ЛІСТИНГ ЗАСТОСУНКУ

Сторінка – 11

Так як лістинг застосунку занадто великий, було прийнято рішення винести сюди найголовніші методи для роботи із процедурами, пошуком даних, , завантаження даних у DataGrid. Оскільки робота з процедурами для кожної таблиці схожа я продемонструю лістинг роботи з однією сутністю.

#### Робота з процедурою для Додавання нового жанру

private void button\_Add\_Set\_categories\_Click(object sender, EventArgs e)

{

using (var sqlConn = new OleDbConnection(connection))

{

var sqlCmd = new OleDbCommand("prosedur\_Add\_categories", sqlConn);

sqlCmd.CommandType = CommandType.StoredProcedure;

sqlCmd.Parameters.AddWithValue("@V\_ID\_Categori ", textBox\_Add\_ID\_Categori\_categories.Text);

sqlCmd.Parameters.AddWithValue("@V\_name\_Categori ", textBox\_Add\_ID\_name\_Categori\_categories.Text);

sqlCmd.Parameters.AddWithValue("@V\_description\_Categori", textBox\_Add\_ID\_description\_Categori\_categories.Text);

sqlConn.Open();

sqlCmd.ExecuteNonQuery();

}

panel\_Menu\_categories.BringToFront();

UpdateSearchAddGrid("categories", dataGridView\_categories);

#### }

#### Робота з процедурою для Редагування жанру

private void button\_Update\_Show\_categories\_Click(object sender, EventArgs e)

{

panel\_Update\_categories.BringToFront();

DataGridViewRow[] dataGridViewRow = new DataGridViewRow[1];

//dataGridView1.Rows.CopyTo()

dataGridView\_categories.SelectedRows.CopyTo(dataGridViewRow, 0);

//dataGridView1.Rows.Remove(row);

textBox\_Update\_ID\_Categori\_categories.Text = dataGridViewRow[0].Cells[0].Value.ToString().Trim();

textBox\_Update\_name\_Categori\_categories.Text = dataGridViewRow[0].Cells[1].Value.ToString().Trim();

textBox\_Update\_description\_Categori\_categories.Text = dataGridViewRow[0].Cells[2].Value.ToString().Trim();

}

private void button\_Update\_Set\_categories\_Click(object sender, EventArgs e)

{

DataGridViewRow[] dataGridViewRow = new DataGridViewRow[1];

dataGridView\_categories.SelectedRows.CopyTo(dataGridViewRow, 0);

var str = dataGridViewRow[0].Cells[0].Value.ToString().Trim();

using (OleDbConnection sqlConn = new OleDbConnection(connection))

{

var sqlCmd = new OleDbCommand("prosedur\_Update\_categories", sqlConn);

sqlCmd.CommandType = CommandType.StoredProcedure;

sqlCmd.Parameters.AddWithValue("@V\_ID\_Categori ", str);

sqlCmd.Parameters.AddWithValue("@V\_name\_Categori ", textBox\_Update\_name\_Categori\_categories.Text.Trim());

sqlCmd.Parameters.AddWithValue("@V\_description\_Categori", textBox\_Update\_description\_Categori\_categories.Text.Trim());

sqlConn.Open();

sqlCmd.ExecuteNonQuery();

}

panel\_Menu\_categories.BringToFront();

UpdateSearchAddGrid("categories", dataGridView\_categories);

}

#### Робота з процедурою для Видалення жанру

private void button\_Delete\_categories\_Click(object sender, EventArgs e)

{

DataGridViewRow[] dataGridViewRow = new DataGridViewRow[1];

dataGridView\_categories.SelectedRows.CopyTo(dataGridViewRow, 0);

var str = dataGridViewRow[0].Cells[0].Value.ToString().Trim();

using (var sqlConn = new OleDbConnection(connection))

{

var sqlCmd = new OleDbCommand("prosedur\_Delete\_categories", sqlConn);

sqlCmd.CommandType = CommandType.StoredProcedure;

sqlCmd.Parameters.AddWithValue("@V\_ID\_Categori ", str);

sqlConn.Open();

sqlCmd.ExecuteNonQuery();

}

UpdateSearchAddGrid("categories", dataGridView\_categories);

#### }

#### Робота з запитом для Пошук жанру

private void button\_Search\_Set\_categories\_Click(object sender, EventArgs e)

{

string sql = "SELECT \* FROM categories";

if (radioButton\_ID\_Categori\_categories.Checked == true)

{

sql = $"SELECT \* FROM categories WHERE ID\_Categori LIKE '%{textBox\_Search\_categories.Text}%'";

}

else if (radioButton\_name\_Categori\_categories.Checked == true)

{

sql = $"SELECT \* FROM categories WHERE name\_Categori LIKE '%{textBox\_Search\_categories.Text}%'";

}

else if (radioButton\_description\_Categori\_categories.Checked == true)

{

sql = $"SELECT \* FROM categories WHERE description\_Categori LIKE '%{textBox\_Search\_categories.Text}%'";

}

else

{

sql = "SELECT \* FROM categories";

}

using (OleDbConnection conn = new OleDbConnection(connection))

{

using (OleDbDataAdapter adapter = new OleDbDataAdapter(sql, conn))

{

DataSet ds = new DataSet();

adapter.Fill(ds);

dataGridView\_categories.DataSource = ds.Tables[0];

}

}

panel\_Menu\_categories.BringToFront();

textBox\_Search\_categories.Text = "";

#### }

#### Робота з запитом для оновлення даних жанру

void UpdateSearchAddGrid(string table, DataGridView dataGridView)

{

string sql = $"SELECT \* FROM {table}";

using (OleDbConnection conn = new OleDbConnection(connection))

{

using (OleDbDataAdapter adapter = new OleDbDataAdapter(sql, conn))

{

DataSet ds = new DataSet();

adapter.Fill(ds);

dataGridView.DataSource = ds.Tables[0];

}

}

#### }

Данна функція викликається майже після кожного натиснення на будь яку клавішу для динамічного оновлення при виводі даних у DataGrid.

#### Код для виводу даних у вкладці жанри

if (tabMain.SelectedIndex == 2)

{

string sql = $"SELECT \* FROM categories";

using (OleDbConnection conn = new OleDbConnection(connection))

{

using (OleDbDataAdapter adapter = new OleDbDataAdapter(sql, conn))

{

DataSet ds = new DataSet();

adapter.Fill(ds);

dataGridView\_categories.DataSource = ds.Tables[0];

}

}

}