Лабораторная работа №4. Работа с базами данных MySQL

Введение

Компания ООО «Мурманрыб» занимается реализацией рыбной продукции в торговые точки города. Компания обладает развитой сетью заказчиков.

Существует несколько категорий продукции: охлажденная рыба, замороженная рыба, полуфабрикаты и готовая продукция. Продукция поступает с рыболовных суден, принадлежащих компании. Некоторые из суден оборудованы рыбоперерабатывающими фабриками для производства филе и прочих полуфабрикатов.

Для привлечения новых заказчиков и повышения продаж компания проводит стимулирующие акции. Акции заключаются в предоставлении сидки на выделенные позиции из ассортимента или в продаже определенного количества позиции по цене меньшего. Каждая акция распространяется только на указанный товар из ассортимента. При этом в одной продаже могут быть учтены акции по нескольким позициям.

Компания ООО «Мурманрыб» разместила заказ на разработку информационной системы, которая должна обеспечить хранение информации об ассортименте и текущем количестве позиций на складе, учет поставок, списания и продаж с учетом действующих акций.

Ассортимент

Ассортимент товара представляет собой полный список товаров, когда-либо продаваемых компанией. У каждого товара учитывается срок хранения, обеспечивающий возможность своевременного списания просроченного товара.

Склад

На склад позиции поступают непосредственно с суден за определенную дату. Стоимость товаров, поставленных в разные дни может отличаться. Пользователь информационной системы должен в любой момент иметь возможность получить информацию о количестве товаров на складе.

Списание товаров со склада может производиться по ряду причин в любом количестве от исходной партии. Списание товаров по причине истечения срока годности производится ежедневно по требованию пользователя.

Продажи

Продажи товаров осуществляются по договору с заказчиком. Каждая продажа может включать в себя несколько позиций, имеющихся в данный момент на складе. В любой момент должен быть обеспечен доступ к продажам за прошедший период с указанием итоговой стоимости и проведенных акций.

Часть I. Создание базы данных

Создайте базу данных в среде MySQL Workbench используя ERD-диаграмму и словарь данных (**MurmanRybERD.pdf** и **MurmanRybDataDictionary.pdf**). Заполните базу данных.

Часть II. Главное окно. Авторизация

1. Создайте новый проект Windows Forms.
2. Добавьте к проекту ссылку на библиотеку MySQL:
   1) выберите пункт меню **Проект – Добавить ссылку…**;
   2) перейдите на вкладку **Сборка – Расширения**;
   3) выберите библиотеку **MySql.Data.dll**;
   4) нажмите кнопку **OK**.
3. Добавьте к проекту класс, например, **DBConnection**, в котором будут содержаться методы для работы с базой данных.
4. В созданном классе подключите пространство имен **MySql.Data.MySqlClient**.
5. Там же объявите следующие статические элементы:

```
static string connectionString = @"Database = MurmanRyb; Data Source = localhost;
                                UserID = root; Password = qwerty "; //строка подключения
static MySqlConnection msConnect; //объект для установки соединения с БД
static MySqlCommand msCommand;    //объект для выполнения запросов
static public MySqlDataAdapter msDataAdapter;
```

6. Далее там же опишите методы установки и отключения соединения с базой данных:

```
//установка соединения с БД
static public bool Connect()
{
   try
   {
         //создание объекта соединения с заданной строкой подключения
         msConnect = new MySqlConnection(connectionString);
         msConnect.Open(); //открытие подключение
         //создание объекта-запрос
         msCommand = new MySqlCommand();
         msCommand.Connection = msConnect;
         msDataAdapter = new MySqlDataAdapter(msCommand);
         return true; //результат «истина»
   }
   catch (Exception ex) //при возникновении ошибки
   {
         //вывод сообщения
         System.Windows.Forms.MessageBox.Show(ex.ToString(), "Ошибка!");
         return false; //результат «ложь»
   }
}

//отключение соединения с БД
static public void Close()
{
   msConnect.Close();
}
```

7. При загрузке главной формы выполните попытку соединения с базой данных:


```
if (!DBConnection.Connect()) //если соединение не установлено
{
   this.Close();             //выход из программы
}
```

8. При закрытии главной формы выполните отключение соединения.

Главное окно программы представляет из себя форму авторизации. Авторизованный пользователь попадает в меню, соответствующее типу его учетной записи.

9. Добавьте к проекту формы **«Меню администратора»** и **«Меню заказчика»**.
10. **В классе работы с БД** объявите следующие открытые поля:

```
static public string User; //логин авторизованного пользователя
static public string Role; //роль авторизованного пользователя
```

11. Там же определите метод авторизации:

```
//авторизация пользователя, принимает параметры с формы авторизации
static public void Authorization(string login, string password)
{
   try
   {
         //формируем запрос: выбрать поле из таблицы значения,
         //где логин и пароль равны введенным пользователем значениям
         string sql = "SELECT Role FROM Users WHERE Login = '" + login
                + "' AND Password = '" + password + "' ;";

         //создаем запрос
         msCommand.CommandText = sql;

         //фиксируем результат запроса
         Object result = msCommand.ExecuteScalar();

         //если в результате выполнения запроса получено непустое значение
         if (result != null)
         {
               //заполняем информацию об авторизованном пользователе
               Role = result.ToString();
               User = login;
         }
         else
         {
               //иначе тип пользователя - неавторизованный
               Role = null;
         }
   }
   catch (Exception ex)      //при возникновении ошибки
   {
         Role = User = null; //обнуляем значения полей
         MessageBox.Show(ex.ToString(), "Ошибка!");
   }
}
```

12. Спроектируйте интерфейс главной формы.
13. Создайте заготовки для форм **«Меню заказчика»** и **«Меню администратора»**.
14. Выставьте ограничения на допустимые значения логина и пароля в соответствующих полях ввода.

15. При нажатии на кнопку **«Авторизация»** выполните проверку наличия учетной записи и при подтверждении переход в соответствующее меню:

```
//вызываем метод авторизации и передаем введенные логин и пароль
DBConnection.Authorization(txtLogin.Text, txtPassword.Text);

switch (DBConnection.Role)
{
   //если роль не распознана, пользователь не авторизован
   case null:
         MessageBox.Show("Неверные данные!");
         break;
   //если авторизован заказчик
   case "customer":
         this.Hide();                                          //скрываем текущую форму
         CustomerMenu CustomerMenuFrm = new CustomerMenu(); //создаем и показываем
         CustomerMenuFrm.Show();                               //меню заказчика
         break;
   //если авторизован администратор
   case "admin":
         this.Hide();                                          //скрываем текущую форму
         AdministratorMenu AdminFrm = new AdministratorMenu();//создаем и показываем
         AdminFrm.Show();                                       //меню администратора
         break;
}
```

16. Разместите в окнах **«Меню заказчика»** и **«Меню администратора»** копки **«Выход».**

17. Реализуйте для кнопок **«Выход»** выход из учетной записи.

Часть III. Просмотр списка пользователей

Окно **«Список пользователей»** предназначено для просмотра в табличном виде текущего списка пользователей и просмотра отфильтрованного списка по типу учетной записи.

1. Создайте форму **«Список пользователей»** и разместите на ней компонент-таблицу типа **DataGrigView**.
2. Обеспечьте переход на форму из **«Меню администратора»**.
3. В модуле функций для работы с базой данных объявите набор данных для хранения информации о пользователях (предварительно подключив пространство имен System.Data):

```
static public DataTable dtUsers = new DataTable();
```

4. Там же объявите метод, позволяющий получить список пользователей для отображения:

```
//метод получения списка пользователей
//selectedRole - значение роли для фильтрации
//по умолчанию = null
static public void GetUserList(string selectedRole = null)
{
   //если роль не выбрана
   if (selectedRole == null)
   {
         //формируем запрос на выборку всех записей
         msCommand.CommandText = "SELECT * FROM Users";
   else
   {
         //иначе, формируем запрос с фильтрацией
         msCommand.CommandText = "SELECT * FROM Users WHERE Users.role='" +
         selectedRole + "'";
   }
   dtUsers.Clear();             //очистка набора данных
   msDataAdapter.Fill(dtUsers); //заполнение набора данных
}
```

5. Включите в метод обработку исключений.
6. В обработчике события загрузки формы **«Список пользователей»** выполните следующие действия:

```
DBConnection.GetUserList();                      //получение списка пользователей
dataGridView1.DataSource = DBConnection.dtUsers; //привязка набора данных к таблице
```

7. Запустите программу и проверьте правильность отображения данных в таблице.
8. Выполните настройку свойств таблицы:
   1) запретите ввод и удаление данных через свойства **AllowUserToAddRows** и **AllowUserToDleteRows**;
   2) в свойстве **Columns** создайте колонки по количеству столбцов в отображаемой таблице;
   3) для каждой колонки установите свойство **DatapropertyName** равным соответствующему имени столбца таблицы БД;
   4) измените подписи заголовков столбцов таблицы на русскоязычные через свойство **HeaderText**.

9. Обеспечьте отбор записей в таблице по полю Роль:
   1) разместите на форме комбинированный список ComboBox;
   2) заполните список значений компонента (Не выбрано, Администратор и Заказчик);
   3) опишите программный код кнопки **«Отбор»**:

```
string selectedRole=null; //выбранное значение поля

//сопоставление номера выбранного значения в списке с типами ролей
switch (comboBox1.SelectedIndex)
{
   case 1:
         selectedRole = "admin";
   break;
   case 2:
         selectedRole = "customer";
   break;
}
DBConnection.GetUserList(selectedRole); //получение списка пользователей
```

10. Запустите программу и проверьте правильность ее работы.

Часть IV. Управление заказчиками

Окно **«Управление заказчиками»** предназначено для просмотра списка заказчиков в табличном виде, добавления и редактирования записей в полях ввода.

1. Создайте форму **«Управление заказчиками»** и спроектируйте ее интерфейс.
2. Выполните отображение данных из таблицы **Заказчики** на форму в компонент типа **DataGridView**.
3. При сохранении новой записи выполните запись данных в таблицы **Заказчики** и **Пользователи** с помощью соответствующих методов:

```
//добавление нового пользователя
static public bool AddUser(string login, string password, string role)
{
   //формирование запроса
   msCommand.CommandText = "INSERT INTO users VALUES('" + login +
                                  "','" + password + "','" + role + "');";
   //выполение запроса
   if (msCommand.ExecuteNonQuery() > 0)
         return true;
   else
         return false;
}

//добавление нового заказчика
static public void AddCustomer(string user, string name, string telephone,  string
   adress, string email = null)
{
   //формирование запроса
   msCommand.CommandText = "INSERT INTO customers VALUES('"+user+"','"+ name
               +"','"+telephone+"','"+email+"','"+adress+"');";
   //выполение запроса
   msCommand.ExecuteNonQuery();
}
```

При добавлении пользователя в базу может возникнуть ошибка. В этом случае необходимо предотвратить добавление записи в таблицу с заказчиками. Одним из вариантов решения данной ситуации является проверка результата работы метода AddUser. Другим вариантом может быть использование транзакции.

Кнопка **«Редактировать»** должна обеспечить изменение данных выделенной пользователем строки таблицы с помощью полей ввода.

18. Разместите компоненты для редактирования записи.
19. Опишите программный код кнопки **«Редактировать»**:

```
//получение значения из 0-й ячейки выделенной строки в таблице
txtLogin.Text = dataGridView1.CurrentRow.Cells[0].Value.ToString();
//...
```

20. Создайте метод обновления записи в базе данных:

```
static public void EditCustomer(string user, string name, string telephone, string
   adress, string email)
{
```



```
        msCommand.CommandText = "UPDATE customers SET name = '" + name +
              "', telephone = '" + telephone +  "', adress='" + adress +
        //выполение запроса
        msCommand.ExecuteNonQuery();
    }
```

21. При сохранении изменений выполните вызов метода редактирования и обновите набор данных.

22. Запустите программу и проверьте правильность ее работы.

23. Дополните форму функцией удаления выделенной записи.

24. Дополните форму функцией отбора записей по указанному пользователем критерию.

Часть V. Управление складом

Администратор имеет возможность просматривать наличие товаров на складе, редактировать позиции ассортимента, производить добавление и списание товаров. В окне **«Управление складом»** отображается текущее состояние склада по каждой позиции (наименование и количество), а также детализация (дата поставки и количество в поставке).

1. Создайте форму **«Управление складом»** и предусмотрите переход на нее из **«Меню администратора»**.
2. Разместите на форме две таблицы.
3. В первую таблицу отобразите информацию о количестве товара на складе для каждого вида. Для формирования содержимого таблицы используйте группирующий запрос, например:

```
msCommand.CommandText = @"SELECT Assortiment.Product, Assortiment.Name,
                                 Sum(Store.Count) AS Count
                          FROM Assortiment
                          INNER JOIN Store USING(Product)
                          WHERE Count>0
                          GROUP BY Product;";
```

4. Во второй таблице отобразите детальную информацию о выбранном в первой таблице товаре:
   1) создайте метод выборки данных, например:

```
msCommand.CommandText = @"SELECT  Store.Date, Store.Count
                            FROM Assortiment
                            INNER JOIN Store USING(Product)
                            WHERE Product = '" + Product + "' ";
```

   2) вызовите данный метод при выборе пользователем строки в первой таблице (обработчик события **CellClick**);
   3) обеспечьте выделение всей строки с помощью мыши (свойство **SelectionMode = FullRowSelect**);
   4) измените цвет выделения строки (свойство **DefaultCellStyle – SelectionBackColor**).
5. Запустите программу и проверьте правильность ее работы.
6. Оформите внешний вид таблиц и заголовки столбцов.
7. Реализуйте добавление товаров на склад:
   1) разместите на форме кнопку **«Добавить поставку»** и поля для ввода информации о поставке;
   2) добавьте метод для добавляющего запроса:

```
static public void AddToStore(string Product, string Count, string Date)
{
   msCommand.CommandText = @"INSERT INTO Store (Product, Count, Date)
                             VALUES('"+Product+"', '"+Count+"','"+Date+"');";
   msCommand.ExecuteNonQuery();
}
```

   3) при нажатии на кнопку **«Добавить поставку»** вызовите соответствующий метод и обновите данные в таблицах.

8. Дополните окно **«Управление складом»** возможностью фильтрации записей по категории товара.

Каждый товар имеет ограниченный срок хранения. Для удобства работы администратора необходимо предусмотреть возможность списания просроченного товара при нажатии на кнопку **«Списание просроченного товара»**.

9. Создайте набор данных для получения информации о товарах с истекшим сроком годности.
10. Создайте метод, позволяющий вносить информацию о списании товара в базу данных:

```
static public int WriteOff()
{
   //формирование запроса на выборку просроченных товаров
   msCommand.CommandText = @"SELECT Store.PositionId, Store.Product, Store.Count,
                                    Store.Date, Assortiment.ShelfLife
                             FROM Store
                             INNER JOIN Assortiment USING(Product)
                             WHERE To_Days(CURDATE())-TO_Days(Store.Date) >=
                                   Assortiment.ShelfLife;";
   dtWriteOff.Clear();
   msDataAdapter.Fill(dtWriteOff); //наполнение набора данных

   //обход таблицы с просроченными товарами
   foreach (DataRow row in dtWriteOff.Rows)
   {
         //преобразование столбца с датой к используемому формату
         DateTime date = new DateTime();
         date = (DateTime)row[3];

         //формирвание запроса на вставку записи в таблицу с просрочкой
         msCommand.CommandText = @"INSERT INTO WriteOff VALUES('" + row[0] + "','"
                                 + row[1] + "','" + row[2] + "','" +
                                 date.ToString(("yyyy-MM-dd")) + "','"+
                                 DateTime.Today.ToString("yyyy-MM-dd") + "')";
         msCommand.ExecuteNonQuery();

         //формирование запроса на удаление позиции со склада
         msCommand.CommandText = @"DELETE FROM Store
                                   WHERE PositionId='" + row[0] + "';";
         msCommand.ExecuteNonQuery();
   }
   return dtWriteOff.Rows.Count; //возвращаем количество просроченных товаров
}
```

11. Реализуйте процесс списания при нажатии на соответствующую кнопку:

```
//выполняем запрос и получем количество просроченных товаров
int count = DBConnection.WriteOff();
if (count > 0)
   MessageBox.Show("Списано " + count.ToString() + " товаров.");
else
   MessageBox.Show("Нет просроченных товаров.");
```

12. Запустите программу и проверьте правильность ее работы.

Часть VI. Задания для самостоятельной работы

1. Используя диаграмму прецедентов (**MurmanRybUseCase.bmp**) дополните недостающий функционал приложения.
2. Спроектируйте структуру базы данных для системы с учетом возможности проведения стимулирующих акций.

