Лабораторная работа № 10.

Тема: Windows Forms. Работа с БД MySQL. Работа с объединением таблиц в базе данных.

Цель: закрепить теоретические знания и применить их на практике.

Задание:

Вариант 1

1. Подготовка к работе.
Задание 1.1:
Создайте базу данных MySQL с именем CompanyDB. В ней создайте две таблицы:
Employees с полями: EmployeeID (первичный ключ, автоинкремент), Name (текст), DepartmentID (число).
Departments с полями: DepartmentID (первичный ключ, автоинкремент), DepartmentName (текст).
Подсказка: Используйте MySQL Workbench или командную строку MySQL для создания базы данных и таблиц.

Задание 1.2:
Добавьте тестовые данные в таблицы:
В таблицу Departments: минимум 3 записи (например, "HR", "IT", "Marketing").
В таблицу Employees: минимум 5 записей, связанных с существующими отделами через поле DepartmentID.
Подсказка: Убедитесь, что значения DepartmentID в таблице Employees соответствуют значениям из таблицы Departments.

Задание 1.3:
Настройте проект Windows Forms для работы с MySQL. Убедитесь, что установлен пакет MySql.Data.
Подсказка: Проверьте наличие пакета в разделе "Dependencies" вашего проекта.

2. Выполнение запросов с объединением таблиц.
Задание 2.1:
Создайте форму с элементом управления DataGridView. При загрузке формы выполните SQL-запрос с использованием INNER JOIN, чтобы вывести список всех сотрудников (Name) и их отделов (DepartmentName).
Подсказка: Используйте запрос:
```
SELECT Employees.Name, Departments.DepartmentName
FROM Employees
INNER JOIN Departments ON Employees.DepartmentID = Departments.DepartmentID;
```

Задание 2.2:
Добавьте возможность выбора отдела из выпадающего списка (ComboBox). После выбора отдела выполните SQL-запрос с использованием WHERE и INNER JOIN, чтобы вывести только сотрудников выбранного отдела.
Подсказка: Используйте параметризованный запрос:
```
SELECT Employees.Name, Departments.DepartmentName
FROM Employees
INNER JOIN Departments ON Employees.DepartmentID = Departments.DepartmentID
WHERE Departments.DepartmentName = @selectedDepartment;
```

Задание 2.3:
Создайте кнопку "Show All Employees". При нажатии на неё выполните SQL-запрос с использованием LEFT JOIN, чтобы вывести всех сотрудников, включая тех, кто не назначен ни в один отдел.
Подсказка: Используйте запрос:
```
SELECT Employees.Name, Departments.DepartmentName
FROM Employees
```

LEFT JOIN Departments ON Employees.DepartmentID = Departments.DepartmentID;

3. Асинхронные запросы с объединением таблиц.
Задание 3.1:
Реализуйте асинхронное выполнение запроса с использованием INNER JOIN при загрузке формы. Во время выполнения запроса покажите индикатор загрузки (например, ProgressBar).
Подсказка: Используйте метод ExecuteReaderAsync().

Задание 3.2:
Добавьте кнопку "Update Department". При нажатии на неё откройте новую форму, где можно выбрать сотрудника из выпадающего списка (ComboBox) и изменить его отдел. Обновите данные в базе данных асинхронно.
Подсказка: Используйте параметризованный запрос для обновления данных:
UPDATE Employees SET DepartmentID = @newDepartmentID WHERE EmployeeID = @employeeID;

4. Агрегирующие функции.
Задание 4.1:
Создайте форму с элементом управления Label. При загрузке формы выполните SQL-запрос с использованием агрегирующей функции COUNT, чтобы вычислить общее количество сотрудников в каждом отделе. Выведите результат в DataGridView.
Подсказка: Используйте запрос:
SELECT Departments.DepartmentName, COUNT(Employees.EmployeeID) AS TotalEmployees
FROM Employees
RIGHT JOIN Departments ON Employees.DepartmentID = Departments.DepartmentID
GROUP BY Departments.DepartmentName;

Задание 4.2:
Добавьте кнопку "Average Employees". При нажатии на неё выполните SQL-запрос с использованием агрегирующей функции AVG, чтобы вычислить среднее количество сотрудников в отделах (исключая пустые отделы). Выведите результат в MessageBox.
Подсказка: Используйте запрос:
```
SELECT AVG(EmployeeCount) AS AverageEmployees
FROM (
   SELECT COUNT(EmployeeID) AS EmployeeCount
   FROM Employees
   GROUP BY DepartmentID
) AS DepartmentCounts;
```

Вариант 2

1. Подготовка к работе.
Задание 1.1:
Создайте базу данных MySQL с именем UniversityDB. В ней создайте две таблицы:
Courses с полями: CourseID (первичный ключ, автоинкремент), CourseName (текст).
Enrollments с полями: EnrollmentID (первичный ключ, автоинкремент), StudentName (текст), CourseID (число).
Подсказка: Используйте MySQL Workbench или командную строку MySQL для создания базы данных и таблиц.

Задание 1.2:
Добавьте тестовые данные в таблицы:
В таблицу Courses: минимум 3 записи (например, "Mathematics", "Physics", "Chemistry").
В таблицу Enrollments: минимум 5 записей, связанных с существующими курсами через поле CourseID.
Подсказка: Убедитесь, что значения CourseID в таблице Enrollments соответствуют значениям из таблицы Courses.

Задание 1.3:
Настройте проект Windows Forms для работы с MySQL. Убедитесь, что установлен пакет MySql.Data.
Подсказка: Проверьте наличие пакета в разделе "Dependencies" вашего проекта.

2. Выполнение запросов с объединением таблиц.
Задание 2.1:
Создайте форму с элементом управления DataGridView. При загрузке формы выполните SQL-запрос с использованием INNER JOIN, чтобы вывести список всех студентов (StudentName) и их курсов (CourseName).
Подсказка: Используйте запрос:
```
SELECT Enrollments.StudentName, Courses.CourseName
FROM Enrollments
INNER JOIN Courses ON Enrollments.CourseID = Courses.CourseID;
```

Задание 2.2:
Добавьте текстовое поле (TextBox) для ввода названия курса. После ввода выполните SQL-запрос с использованием WHERE и INNER JOIN, чтобы вывести только студентов, записанных на указанный курс.
Подсказка: Используйте параметризованный запрос:
```
SELECT Enrollments.StudentName, Courses.CourseName
FROM Enrollments
INNER JOIN Courses ON Enrollments.CourseID = Courses.CourseID
WHERE Courses.CourseName = @courseName;
```

Задание 2.3:
Создайте кнопку "Show Unassigned Students". При нажатии на неё выполните SQL-запрос с использованием LEFT JOIN, чтобы вывести всех студентов, включая тех, кто не записан ни на один курс.
Подсказка: Используйте запрос:
```
SELECT Enrollments.StudentName, Courses.CourseName
FROM Enrollments
LEFT JOIN Courses ON Enrollments.CourseID = Courses.CourseID;
```

3. Асинхронные запросы с объединением таблиц.
Задание 3.1:
Реализуйте асинхронное выполнение запроса с использованием INNER JOIN при загрузке формы. Во время выполнения запроса покажите анимацию загрузки (например, PictureBox с GIF-изображением).
Подсказка: Используйте метод ExecuteReaderAsync().

Задание 3.2:
Добавьте кнопку "Assign Course". При нажатии на неё откройте новую форму, где можно выбрать студента из выпадающего списка (ComboBox) и назначить ему курс. Обновите данные в базе данных асинхронно.
Подсказка: Используйте параметризованный запрос для обновления данных:
```
UPDATE Enrollments SET CourseID = @newCourseID WHERE StudentName = @studentName;
```
Эти задания помогут студентам освоить работу с объединением таблиц в контексте компании (CompanyDB) и университета (UniversityDB). Они включают создание, выполнение и асинхронную обработку запросов с использованием INNER JOIN, LEFT JOIN и других типов объединений.

4. Агрегирующие функции.
Задание 4.1:
Создайте форму с элементом управления Label. При загрузке формы выполните SQL-запрос с использованием агрегирующей функции COUNT, чтобы вычислить общее количество студентов, записанных на каждый курс. Выведите результат в DataGridView.
Подсказка: Используйте запрос:
```
SELECT Courses.CourseName, COUNT(Enrollments.EnrollmentID) AS TotalStudents
```

```
FROM Enrollments
RIGHT JOIN Courses ON Enrollments.CourseID = Courses.CourseID
GROUP BY Courses.CourseName;
```

Задание 4.2:
Добавьте кнопку "Most Popular Course". При нажатии на неё выполните SQL-запрос с использованием агрегирующей функции MAX, чтобы найти курс с наибольшим количеством записанных студентов. Выведите результат в Label.
Подсказка: Используйте запрос:

```
SELECT CourseName, MAX(StudentCount) AS MaxStudents
FROM (
    SELECT Courses.CourseName, COUNT(Enrollments.EnrollmentID) AS StudentCount
    FROM Enrollments
    RIGHT JOIN Courses ON Enrollments.CourseID = Courses.CourseID
    GROUP BY Courses.CourseName
) AS CourseCounts;
```

Отчет должен содержать (см. образец):
- номер и тему лабораторной работы;
- фамилию, номер группы студента и вариант задания;
- скриншоты окна Visual Studio с исходным кодом программ и комментариями;
- скриншоты с результатами выполнения программ;
- пояснения, если необходимо;
- выводы.

Отчеты в формате **pdf** отправлять на email: **colledge20education23@gmail.com**