МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

**Национальный исследовательский
Нижегородский государственный университет им. Н.И. Лобачевского**

С.П. Никитенкова

РАЗРАБОТКА WPF-ПРИЛОЖЕНИЙ НА ОСНОВЕ БАЗ ДАННЫХ

Учебно-методическое пособие

Рекомендовано методической комиссией радиофизического факультета
для студентов ННГУ, обучающихся по специальностям
10.05.02 «Информационная безопасность телекоммуникационных систем» и
02.03.02 «Фундаментальная информатика и информационные технологии»

Нижний Новгород

2019

УДК 004.514.62
ББК 32.973.2-018
Н62

Никитенкова С.П. Разработка WPF-приложений на основе баз данных: Учебно-методическое пособие. – Нижний Новгород: Нижегородский госуниверситет, 2019. – 46 с.

Рецензент: к.ф.-м.н., доцент **Жуков С.Н.**

Учебно-методическое пособие представляет собой начальный курс по изучению возможностей технологии Windows Presentation Foundation (WPF) при разработке пользовательских интерфейсов для эффективной работы с базами данных. На практических примерах рассматривается процесс создания и реализации пользовательских приложений, решающих типовые задачи манипулирования данными. Рассматривается взаимосвязь декларативного языка расширенной разметки для приложений XAML и базовых компонент WPF. Пособие содержит контрольные вопросы и задания, которые могут быть использованы в процессе обучения.
Пособие предназначено для студентов, обучающихся по специальностям 10.05.02 «Информационная безопасность телекоммуникационных систем» и 02.03.02 «Фундаментальная информатика и информационные технологии».

Ответственный за выпуск:
зам. председателя методической комиссии радиофизического факультета ННГУ,
д.ф.-м.н., профессор **Е.З. Грибова**

УДК 004.056.53
ББК 32.973.2-018

СОДЕРЖАНИЕ

Введение

Базы данных применяются сегодня практически во всех приложениях, начиная от мобильных и Web-приложений и заканчивая системами управления предприятия.

База данных — совокупность связанных данных, организованных по определенным правилам, предусматривающим общие принципы описания, хранения и манипулирования данными. База данных – это именованная совокупность данных, отражающая состояние объектов и их взаимосвязей (отношений) в рассматриваемой предметной области.

Система управления базами данных (СУБД) – это программный комплекс, обеспечивающий функционирование базы данных. Он отвечает за сохранность, безопасность, целостность, взаимное соответствие данных и обеспечивает доступ пользователей к данным. Система управления базами данных представляет собой совокупность языковых и программных средств, предназначенных для создания, ведения и совместного использования БД многими пользователями.

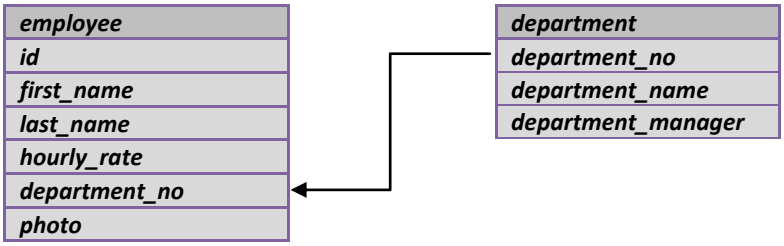
Существует много различных стратегий организации данных для облегчения доступа и манипулирования ими. На сегодняшний день наиболее популярными являются реляционные базы данных. Реляционной называется база данных, в которой все данные, доступные пользователю, организованы в виде взаимосвязанных таблиц, а все операции базы данных выполняются над этими таблицами. Некоторые популярные системы управления реляционными базами данных (RDBMS) - это Microsoft SQL Server, Oracle, PostgreSQL и MySQL и т.д.

SQL (Structured Query Language) – это язык структурированных запросов к реляционным базам данных. На этом языке можно формулировать запросы, которые извлекают данные, удовлетворяющие заданным критериям, модифицируют данные. На языке SQL можно создавать таблицы, изменять их структуры, определять права доступа к данным и т.д. Стандарт SQL поддерживается всеми ведущими мировыми фирмами, действующими в сфере технологий баз данных.

Операции с базой данных выполняются с помощью специальных приложений, предоставляющих пользователю удобный интерфейс. Приложения пишутся на языках программирования, таких как C, Java, Python и т.д. Использование выразительного и эффективного стандартного языка SQL позволяет обеспечить высокую степень независимости разрабатываемых прикладных программных систем от конкретного типа используемой СУБД, существенно поднять уровень и унификацию инструментальных средств разработки приложений, работающих с реляционными базами данных

Базовые инструкции языка SQL

SQL применяется не только для работы с табличными данными, но и для выполнения всех операций с базами данных, включая создание и модификацию таблиц. Пусть база данных содержит две таблицы Сотрудник (Employee) и департаменты (Department).



Оператор CREATE TABLE

Чтобы создать таблицу с помощью инструкции CREATE TABLE необходимо указать следующие данные: имя новой таблицы, которое задается после ключевых слов CREATE TABLE, имена и определения столбцов таблицы, разделенные запятыми. Таблица employee может быть создана

```
CREATE TABLE employee ( id int(10) primary key auto_increment,
                first_name varchar(20), last_name varchar(20),
                hourly_rate decimal (10,2), department_no char(4), photo longblob);
```

Столбец id является полем первичного ключа – столбцом, значения которого уникально идентифицируют каждую строку таблицы employee.

Оператор ALTER TABLE

Для того чтобы обновить определение таблицы следует воспользоваться инструкцией ALTER TABLE. Столбец с данными о должностях, занимаемых сотрудниками, может быть добавлен следующим образом

```
ALTER  TABLE employee ADD position VARCHAR(20);
```

При должном уровне анализа будущих потребностей пользователей структура таблицы не должны меняться после того, как в таблицу введены данные.

Ссылочная целостность данных

Внешний ключ SQL — это ключ, используемый для объединения двух таблиц. Иногда его также называют ссылочным ключом. Внешний ключ — это столбец или комбинация столбцов, значения которых соответствуют Первичному ключу PRIMARY KEY в другой таблице.

Связь между двумя таблицами задается через соответствие Первичного ключа в одной из таблиц внешнему ключу во второй. Ограничение FOREIGN KEY используется в команде CREATE TABLE (или ALTER TABLE), содержащей поле, которое объявлено внешним ключом.

Например, между таблицами employee и department может быть установлена связь по полю department_no. SQL поддерживает ссылочную целостность с ограничением FOREIGN KEY. Сотрудника нельзя зачислить в несуществующий отдел – в таблицу employee нельзя вести значения department_no, отсутствующие в таблице department. В таблице department нельзя удалить значения department_no, если они присутствуют в поле department_no таблицы employee.

Внешний ключ FOREIGN KEY может быть добавлен следующим образом:

```
ALTER  TABLE employee ADD FOREIGN KEY (department_no)
                        REFERENCES department(department_no);
```

Оператор INSERT

Простейший способ добавления данных в таблицу реализуется с помощью базового синтаксиса инструкции INSERT. Для этого нужно указать имя таблицы и значения, которые должны быть введены в новую строку. В самой простой форме, INSERT использует следующий синтаксис:

```
INSERT INTO <table name> VALUES ( <value>, <value> . . .);
```

Чтобы ввести строку в таблицу Сотрудник (employee) необходимо (дробная часть числа отделяется от его целой части точкой):

```
INSERT INTO employee VALUES ('Paul', 'Deitel',10.12, 'L004');
```

Данные, которые должны быть сохранены в каждом столбце таблицы, указываются в предложении VALUES. Если имена столбцов отсутствуют, должно

быть приведено значение для каждого столбца таблицы. Если для какого-то столбца нет соответствующего значения, следует указать NULL (предполагается, что таблица допускает отсутствие значений в этих столбцах). Значения столбца id не указываются, т.к. для него выбран режим auto_increment. Столбцы должны заполняться в порядке, в котором они перечислены в определении таблицы. Когда строка вводится в таблицу, СУБД устанавливает соответствие каждого элемента в списке столбцов с соответствующим значением в списке VALUES. Первое значение в списке VALUES соответствует первому указанному имени столбца, второе значение – второму имени и т.д. Каждое значение в предложении значений VALUES должно совпадать с типом данных столбца таблицы.

Имена столбцов могут быть перечислены в явном виде, тогда значения, указанные в предложении VALUES, должны соответствовать им в том же самом порядке, причем порядок перечисления полей не обязательно должен совпадать с порядком столбцов в реальной таблице. Например,

```
INSERT INTO employee (last_name, first_name) VALUES ('Deitel', 'Paul');
```

Оператор UPDATE

Для обновления (модификации) данных какой-либо таблицы предназначена инструкция UPDATE, которую можно использовать для обновления определенных строк в таблице, а также для обновления всех строк в таблице.

Инструкция UPDATE состоит из трех основных частей: имя таблицы, подлежащей обновлению; имена столбцов и их новые значения; условия фильтрации, определяющие, какие именно строки должны быть обновлены.

Рассмотрим пример. Допустим, у сотрудника с id 1001 увеличилась оплата за час (hourly_rate) и/или он переведен в другой департамент, поэтому запись с id 1001 нужно обновить. Такое обновление можно выполнить посредством следующей инструкции.

```
 UPDATE employee SET hourly_rate =50.34 WHERE id = 1001;

UPDATE employee SET hourly_rate =50.34, department_no='L028'
            WHERE id = 1001;
```

Если часть WHERE id = 1001 будет отсутствовать, то значение hourly_rate =50.34 будет установлено во всех записях таблицы, т.е. для всех сотрудниках.

Оператор DELETE

Для удаления данных из таблицы предназначена инструкция DELETE. Ее можно использовать для удаления определенных строк из таблицы; для удаления всех строк из таблицы. Следующая инструкция удаляет одну строку из таблицы employee.

```
DELETE  FROM employee WHERE id = 1001;
```

Если бы предложение WHERE пропущено, инструкция удаляет все строки из таблицы. Инструкция DELETE никогда не удаляет саму таблицу.

При использовании инструкций UPDATE или DELETE в предложении WHERE рекомендуется использовать первичный ключ таблицы всякий раз, когда это возможно. Прежде чем использовать предложение WHERE с инструкцией UPDATE или DELETE, необходимо проверить его с инструкцией SELECT, дабы убедиться в том, что оно правильно фильтрует записи, т.к. можно ошибиться и записать неправильное условие. Необходимо использовать внешние ключи, чтобы СУБД не позволяла удалять строки, для которых в других таблицах имеются связанные с ними данные.

Оператор SELECT

SELECT - оператор запроса в языке SQL, возвращающий набор данных (выборку) из базы данных.

Для извлечения данных из нескольких столбцов таблицы применяется инструкция SELECT, в которой необходимо через запятую указать имена столбцов.

```
SELECT first_name, last_name, hourly_rate FROM employee;
```

Помимо извлечения указанных столбцов (одного или нескольких), с помощью инструкции SELECT можно запросить все столбцы, не перечисляя каждый из них. Для этого вместо имен столбцов указывается групповой символ «звездочка» (*).

```
SELECT * FROM employee;
```

Ключевое слово DISTINCT в запросе заставляет СУБД вернуть только уникальные значения. Предложение SELECT DISTINCT department_no заставляет СУБД вернуть только записи с отличающимися значениями department_no.

```
SELECT DISTINCT department_no FROM employee;
```

Инструкция SELECT возвращает все строки, соответствующие критерию отбора. Если необходимо получить лишь первую строку или заданное число строк, то в СУБД MySQL, MariaDB и SQLite можно воспользоваться предложением LIMIT.

```
SELECT first_name, last_name, hourly_rate FROM employee LIMIT 5;
```

Для сортировки извлекаемых инструкцией SELECT данных предназначено предложение ORDER BY. В нем указывается имя одного или нескольких столбцов, по которым и сортируются результаты запроса. Рассмотрим следующий пример. Приведенный ниже запрос позволяет отсортировать результат запроса в порядке возрастания значения оплаты за час сотрудника.

```
SELECT last_name, hourly_rate FROM employee ORDER BY hourly_rate;
```

Приведенный ниже запрос позволяет получить данные о самом низкооплачиваемом сотруднике

```
SELECT first_name, last_name, hourly_rate FROM employee
ORDER BY first_name, last_name LIMIT 1;
```

Данные можно сортировать не только по возрастанию (в алфавитном порядке). Такой порядок задан по умолчанию, но в предложении ORDER BY можно также указывать порядок по убыванию. Для этого предназначено ключевое слово DESC (сокращение от DESCENDING).

```
SELECT first_name, last_name, hourly_rate FROM employee
ORDER BY hourly_rate DESC;
```

Данные можно отсортировать по нескольким столбцам. Например, список сотрудников сортируется сначала по фамилии, затем по имени.

```
SELECT first_name, last_name, hourly_rate FROM employee
ORDER BY first_name, last_name;
```

В таблицах баз данных обычно содержится очень много информации, и довольно редко возникает необходимость извлекать все строки из таблицы. Гораздо чаще требуется извлечь какую-то часть данных для последующей обработки или составления отчетов. В этом случае необходимо указать критерий отбора, также называемый условием фильтрации.

В инструкции SELECT данные фильтруются путем указания критерия отбора в предложении WHERE. Это предложение указывается сразу после названия таблицы (в предложении FROM) следующим образом:

```
SELECT first_name, last_name, hourly_rate FROM employee
WHERE hourly_rate =3.49;
```

Условие WHERE может включать в себя предикаты AND, OR, NOT, LIKE, BETWEEN, IS, IN, ключевое слово NULL, операторы сравнения и равенства (<, >, =).

В следующем примере демонстрируется применение оператора BETWEEN для извлечения всех сотрудников, оплата которых выше 5 и ниже 10 за час.

```
SELECT first_name, last_name, hourly_rate FROM employee
WHERE hourly_rate BETWEEN 5 AND 10;
```

Тот же запрос может быть записан иначе

```
SELECT first_name, last_name, hourly_rate FROM employee
WHERE ((hourly_rate>= 5) AND (hourly_rate<=10));
```

Чтобы проверить, содержит ли столбец значение NULL, нельзя просто записать «= NULL». Для предложения WHERE предусмотрено специальное выражение, которое используется для проверки значений NULL в столбцах: IS NULL. Синтаксис выглядит следующим образом:

```
SELECT first_name, last_name, hourly_rate FROM employee
WHERE hourly_rate IS NULL;
```

Предикат IN служит для указания диапазона условий, любое из которых может быть выполнено. При этом значения, заключенные в скобки, перечисляются через запятую.

```
SELECT first_name, last_name, hourly_rate FROM employee
WHERE department_no id IN ('L004', 'L028') ORDER BY last_name;
```

Благодаря метасимволам можно создавать расширенные условия отбора строк. В данном примере, для того чтобы найти всех сотрудников, фамилия которых начинается на букву 'A', необходимо составить соответствующий шаблон поиска. Наиболее часто используемый метасимвол — знак процента (%). В шаблоне поиска знак % означает найти все вхождения любого символа.

```
SELECT first_name, last_name, hourly_rate FROM employee
WHERE last_name LIKE 'A%';
```

Шаблон ' % stein % ' означает найти всех сотрудников, фамилия которых содержит текст stein в любом месте, независимо от количества символов перед указанным текстом или после него. В качестве подстановочного знака может использовать символ '_'. Шаблон '_ instein' означает найти всех сотрудников, фамилия которых содержит текст instein и отличается первой буквой.

Часто бывает необходимо подвести итоги, не отображая исходные данные. В SQL для этого предусмотрены специальные функции. Запросы с такими функциями часто используются для анализа данных и создания отчетов. Примерами подобных запросов могут служить: подсчет числа строк в таблице (либо числа строк, которые удовлетворяют какому-то условию или содержат определенное значение); определение суммы по набору строк в таблице; поиск наибольшего, наименьшего и среднего значений в столбце таблицы (по всем или каким-то конкретным строкам).

В каждом из этих примеров пользователю нужны итоговые сводки по таблице, а не исходные данные. Чтобы облегчить извлечение подобной информации, в SQL предусмотрен набор из пяти итоговых функций AVG(), COUNT(), MAX(), MIN(), SUM(). Например, средняя оплата за час может быть найдена как

```
SELECT AVG(hourly_rate) FROM employee;
```

Группировка дает возможность разделить все данные на логические наборы, благодаря чему становится возможным выполнение статистических вычислений

отдельно по каждой группе. Группы создаются с помощью предложения GROUP BY инструкции SELECT.

Например, необходимо узнать количество сотрудников в каждом отделе,

```
SELECT department_no, COUNT(id) as employee_ number FROM employee
GROUP BY department_no;
```

и среднюю оплату за час в каждом отделе.

```
SELECT department_no, AVG(hourly_rate) FROM employee
GROUP BY department_no;
```

SQL позволяет не только группировать данные с помощью предложения GROUP BY, но и осуществлять их фильтрацию, т.е. указывать, какие группы должны быть включены в результаты запроса, а какие — исключены из них. Например, необходимо узнать среднюю оплату за час в каждом отделе с числом сотрудников больше 20. Чтобы получить такие данные, необходим фильтр, относящийся к целой группе, а не к отдельным строкам.

```
SELECT department_no, AVG(hourly_rate) FROM employee
GROUP BY department_no HAVING COUNT(id)>=20;
```

Подзапросы

Подзапрос — форма команды SELECT, которая появляется внутри другого оператора SQL. Подзапрос иногда называется вложенным запросом. Подзапросы могут использоваться для следующих целей: для определения множества строк, вставляемых в целевую таблицу операторами INSERT или CREATE TABLE; для определения одного или более значений, назначаемых существующим строкам в операторе UPDATE; для обеспечения необходимых условий в выражениях WHERE, HAVING операторов SELECT, UPDATE, и DELETE.

Например, данные о самом высокооплачиваемом сотруднике могут быть получены как результат запроса

```
SELECT first_name, last_name, hourly_rate FROM employee
where hourly_rate=(select max(hourly_rate) from employee);
```

Объединение таблиц

В рассматриваемом примере созданы две таблицы: одна — для хранения информации о сотрудниках, другая — о департаментах. Данные из обеих таблиц могут быть извлечены посредством объединений. Объединение представляет собой механизм слияния таблиц в инструкции SELECT, позволяющий объединить несколько исходных таблиц в одну общую, которая будет связывать нужные строки из каждой таблицы.

```
SELECT id, first_name, last_name, hourly_rate, department_name,
department_manager FROM employee INNER JOIN department
ON employee.department_no = department.department_no;
```

Столбцы id, first_name, last_name, hourly_rate находятся в одной таблице, department_name, department_manager − в другой. SQL не ограничивает число таблиц, которые могут быть объединены посредством инструкции SELECT..

Разработка клиентского приложения

Open Database Connectivity (ODBC) - это стандартный интерфейс прикладного программирования (API) для доступа к системам управления базами данных (СУБД). Достоинство технологии ODBC заключается в простоте разработки приложений, обусловленной высоким уровнем абстрактности интерфейса доступа к данным практически любых существующих типов СУБД. Язык SQL используется как основной стандарт доступа к данным. Интерфейс ODBC обеспечивает высокую степень универсальности: одно приложение может обращаться к разным SQL-совместимым СУБД посредством общего кода.

WPF является рекомендуемой технологией реализации пользовательских интерфейсов для Windows-приложений. Windows Presentation Foundation (WPF) представляет собой обширный API-интерфейс для создания настольных графических программ имеющих насыщенный дизайн и интерактивность. Windows Presentation Foundation (WPF) — это платформа пользовательского интерфейса для создания клиентских приложений для настольных систем. Платформа разработки WPF поддерживает широкий набор компонентов для разработки приложений, включая модель приложения, ресурсы, элементы управления, графику, макет, привязки данных, документы и безопасность. WPF использует расширяемый язык разметки для приложений (XAML), чтобы предоставить декларатив-

ную модель для программирования приложений. XAML (eXtensible Application Markup Language) – это язык разметки, основанный на XML, являющимся де факто стандартом для обмена данными. XAML представляет собой язык разметки, используемый для создания экземпляров объектов .NET. Хотя язык XAML – это технология, которая может быть применима ко многим различным предметным областям, его основное назначение – конструирование пользовательских интерфейсов WPF. Документы XAML определяют расположение панелей, кнопок и прочих элементов управления, составляющих окна в приложении WPF.

Рассмотрим разработку простейшего WPF-приложений, сочетающего XAML и процедурный код. Приложение обеспечивает чтение и запись данных в таблицы, рассмотренные в части I настоящего пособия. База данных организована средствами СУБД MySQL.

В приложении будут рассматриваться две разновидности таблицы employee, хранящей информации о сотрудниках. В таблице employee фотография сотрудника храниться в отдельной папке, а в таблицу записывается путь к этому файлу. В таблице employee_photo фотография сотрудника храниться непосредственно в таблице в виде данных типа longblob.

```
mysql> use my_company;
Database changed
mysql> desc employee;
+---------------+---------------+------+-----+---------+----------------+
| Field         | Type          | Null | Key | Default | Extra          |
+---------------+---------------+------+-----+---------+----------------+
| id            | int(11)       | NO   | PRI | NULL    | auto_increment |
| first_name    | varchar(20)   | YES  |     | NULL    |                |
| last_name     | varchar(20)   | YES  |     | NULL    |                |
| hourly_rate   | decimal(10,2) | YES  |     | NULL    |                |
| department_no | varchar(7)    | YES  |     | NULL    |                |
| photo         | varchar(50)   | YES  |     | NULL    |                |
+---------------+---------------+------+-----+---------+----------------+
6 rows in set (0.09 sec)
```

```
mysql> desc employee_photo;
+---------------+---------------+------+-----+---------+----------------+
| Field         | Type          | Null | Key | Default | Extra          |
+---------------+---------------+------+-----+---------+----------------+
| id            | int(10)       | NO   | PRI | NULL    | auto_increment |
| first_name    | varchar(20)   | YES  |     | NULL    |                |
| last_name     | varchar(20)   | YES  |     | NULL    |                |
| hourly_rate   | decimal(10,2) | YES  |     | NULL    |                |
| department_no | char(4)       | YES  |     | NULL    |                |
| photo         | longblob      | YES  |     | NULL    |                |
+---------------+---------------+------+-----+---------+----------------+
6 rows in set (0.00 sec)
```

```
mysql> desc department;
+-------------------+-------------+------+-----+---------+-------+
| Field             | Type        | Null | Key | Default | Extra |
+-------------------+-------------+------+-----+---------+-------+
| department_no     | char(4)     | NO   | PRI | NULL    |       |
| department_name   | varchar(20) | YES  |     | NULL    |       |
| department_manager| varchar(20) | YES  |     | NULL    |       |
+-------------------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

При создании приложения одновременно с отображением xaml-файла для класса окна выводится окно визуального редактора. По умолчанию в окно приложения уже включен группирующий компонент Grid – наиболее универсальный из группирующих компонентов, позволяющий размещать свои дочерние компоненты в нескольких строках и столбцах. Кроме того, для класса MainWindow определены три свойства: Title, Height и Width. Гораздо большее число свойств приведено в окне Properties.

В файле MainWindow.xaml.cs содержится частичное определение класса MainWindow, включающее конструктор без параметров, в котором вызывается метод InitializeComponent, обеспечивающий начальную инициализацию всех компонентов окна. Все действия с компонентами можно выполнять только после их начальной инициализации, поэтому пользовательский код добавляется в конструктор после вызова данного метода.

Предварительно в СУБД MySQL создана база данных my_company, содержащая две таблицы employee и department

Создадим проект WPF. Выберем в контекстном меню созданного проекта WpfApplication в Обозревателе решений добавить ссылку на MySql.Data.dll. В проекте добавляется пространство имен:

```
using MySql.Data.MySqlClient;
using System.Data;
using MySql.Data;
```

Создание подключения к базе данных:

```
string Connect="Database=БАЗА;Data Source=ХОСТ;
                User Id=ПОЛЬЗОВАТЕЛЬ; Password=ПАРОЛЬ";
MySqlConnection myConnection = new MySqlConnection(Connect);

string query="SELECT * FROM employee";

MySqlCommand myCommand = new MySqlCommand(query, myConnection);

myConnection.Open(); //Устанавливаем соединение с базой данных.
//...
//...
myConnection.Close();
```

Рассмотрим простейшее приложение. Создадим класс Employee. При создании нового объекта информация о нем будет заноситься в таблицу employee базы данных my_company. При считывании информации из таблицы employee данные будут отображаться в скроллируемом текстовом поле TextBox1.

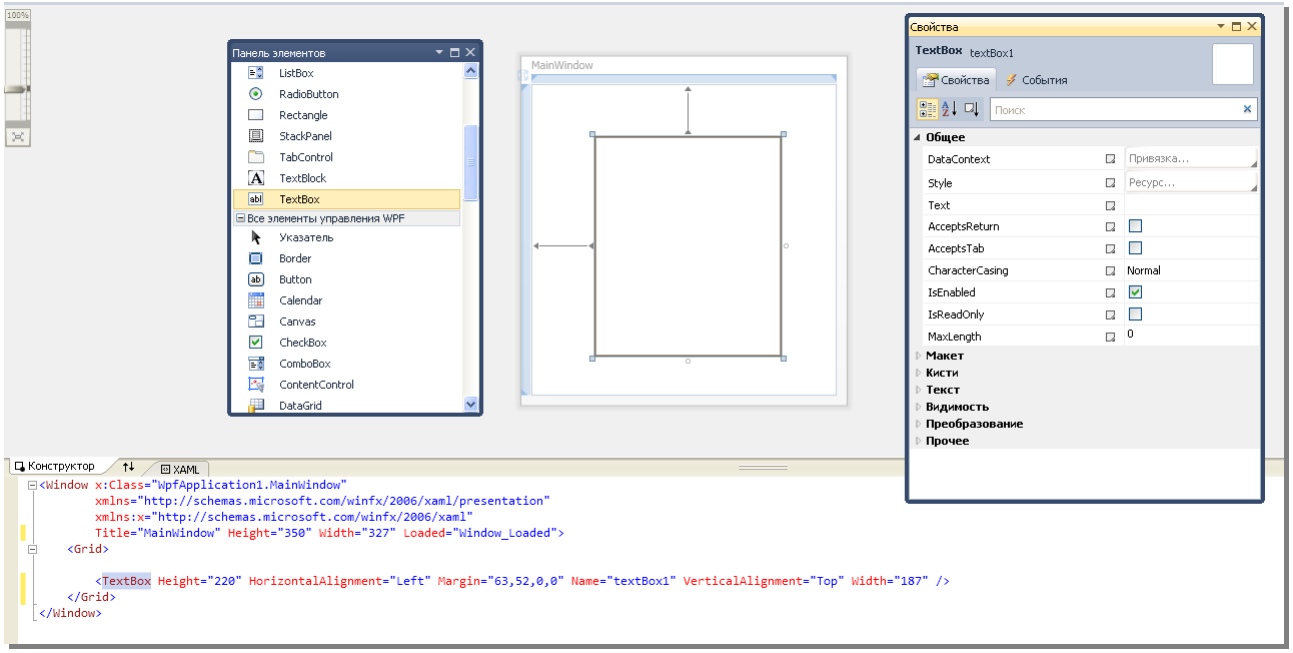


Рис. 1. Разработка программы с помощью визуального редактора

Создаем класс Employee

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace WpfApplication1
{
   class Employee
   {
      private int id;
      private string first_name;
      private string last_name;
      private double hourly_rate;

      public Employee(string first_name, string last_name, double hourly_rate)
      {
```

```
            this.first_name = first_name;
            this.last_name=last_name;
            this.hourly_rate = hourly_rate;

        }

        public void set_ID(int id) { this.id = id; }
        public int get_ID() { return id; }

        public void set_First_Name(string first_name) { this.first_name = first_name; }
        public string get_First_Name() { return first_name; }

        public void set_Last_Name(string last_name) { this.last_name = last_name; }
        public string get_Last_Name() { return last_name; }

        public void set_Hourly_Rate(double hourly_rate) { this.hourly_rate = hourly_rate; }
        public double get_Hourly_Rate() { return hourly_rate; }
    }
}
```

Создаем приложение

```
using MySql.Data.MySqlClient;
using System.Data;
using MySql.Data;

using System.Globalization;

using System.Windows.Threading;
using System.Threading;


namespace WpfApplication1
{

    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
        }

        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            Employee x = new Employee("John", "Smith", 1.1);

// устанавливаем "." как разделить между дробной и целой частью числа
    Thread.CurrentThread.CurrentCulture = new CultureInfo("en-US");
```

```
string Connect = "Database=my_company;Data Source=localhost;User Id=root;Password=pwd ";

 MySqlConnection connection = new MySqlConnection(Connect);

string query = "INSERT INTO employee (first_name, last_name, hourly_rate) VALUES ('"
            + x.get_First_Name() + "','" + x.get_Last_Name() + "'," + x.get_Hourly_Rate() + ");";

      MySqlCommand command = new MySqlCommand(query, connection);
      connection.Open();

      command.ExecuteReader();

      connection.Close();

//  считываем данные

      List<Employee> my_list = new List<Employee>();

      string sql = "SELECT id, first_name, last_name, hourly_rate from employee;";

      MySqlCommand command1 = new MySqlCommand(sql, connection);

      connection.Open();

      MySqlDataReader reader = command1.ExecuteReader();

      double number;

      if (reader.HasRows)
    {
         while (reader.Read())
         {

                  Employee my = new Employee("","",0.0);

                  my.set_ID(Convert.ToInt32(reader["id"]));
                  my.set_First_Name(reader["first_name"].ToString());
                  my.set_Last_Name(reader["last_name"].ToString());

                  if (double.TryParse(reader["hourly_rate"].ToString(), out number))
                     my.set_Hourly_Rate(number);

                   else
                     my.set_Hourly_Rate(0);

              my_list.Add(my);

         }
                   textBox1.ScrollToEnd();
```

```
            foreach(var emp in my_list)

            {
               textBox1.Text = textBox1.Text + "ID:"+emp.get_ID() +
               "\n First Name: " + emp.get_First_Name() +
               "\n Last Name: " + emp.get_Last_Name() +
               "\n Hourly Rate " + emp.get_Hourly_Rate() + "\n";
            };

      }
         else
         {
            MessageBox.Show("No rows found.");
         }

         reader.Close();
      }

   connection.Close();

   }
}
```

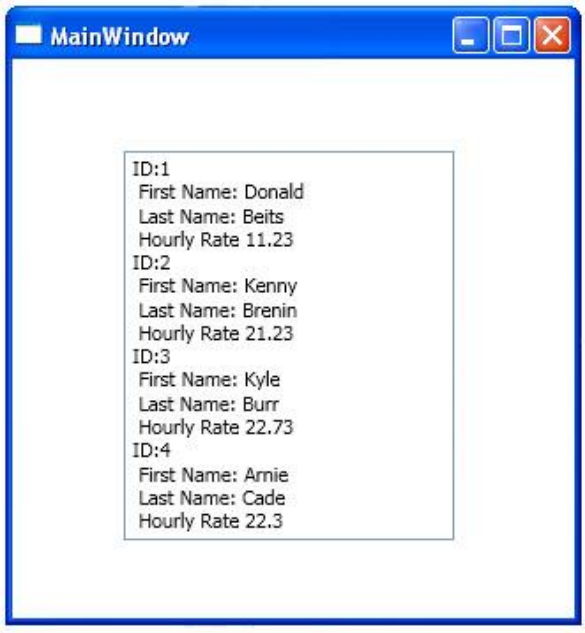


Рис. 2. Результат выполнения программы

Вывод результата запроса может быть реализован через класс DataTable, например,

```
MySqlConnection con= new MySqlConnection(Connect);

 MySqlCommand cmd = con.CreateCommand();

 cmd.CommandText = @"select * from employee limit 10";

 MySqlDataReader rd;

 con.Open();

 rd = cmd.ExecuteReader();

 DataTable employees = new DataTable();

 employees.Load(rd);

 foreach (DataRow row in employees.Rows)
 {
   textBox1.Text = textBox1.Text+ row["first_name"].ToString()+" "+
                   row["last_name"].ToString()+" "+row["hourly_rate"].ToString()+"\n";
 }
```

Запись информации в базу данных

Разработаем более сложное приложение, имеющее более сложный дизайн и позволяющее сохранить фотографию сотрудника.

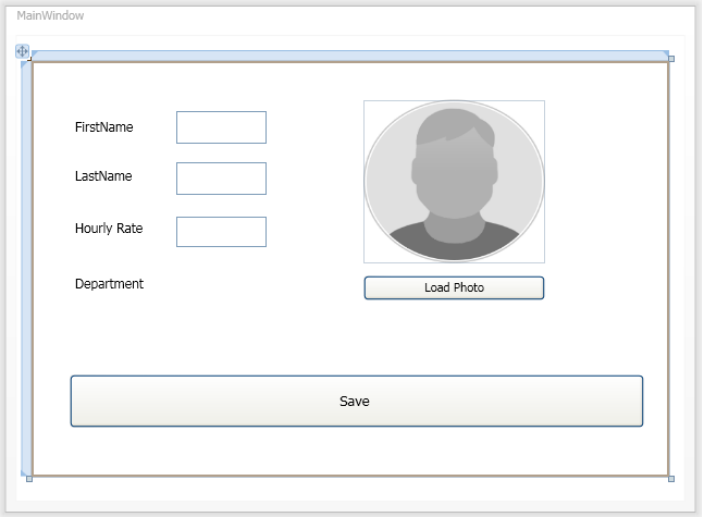


Рис. 3. Окно заполнения данных о сотруднике

С помощью визуального конструктора создадим пользовательский интерфейс, показанный на рис.3. Используем элементы управления текстовые поля

(TextBox), метки (Label), списки (ComboBox), кнопки (Button) и изображения Image Текстовые поля для ввода информации переименованы:

```
<TextBox Height="30" HorizontalAlignment="Left" Margin="133,46,0,0"
  Name="textBox_FirstName" VerticalAlignment="Top" Width="83" MaxLines="20" />

<TextBox Height="30" HorizontalAlignment="Left" Margin="133,93,0,0"
  Name="textBox_LastName" VerticalAlignment="Top" Width="83" MaxLines="20" />

<TextBox Height="28" HorizontalAlignment="Left" Margin="133,143,0,0"
 Name="textBox_HourlyRate" VerticalAlignment="Top" Width="83" MaxLines="20"  />
```

В начале проекта должно быть добавлено пространство имен

```
using MySql.Data.MySqlClient;
using Microsoft.Win32;
using System.IO;
using System.Data;
```

При загрузке окна будет вызван метод Window_Loaded(object sender, RoutedEventArgs e), в котором устанавливаются начальные значения. Данные о департаментах считываются из таблицы department.

```
private void Window_Loaded(object sender, RoutedEventArgs e)
      {

   string Connect = "Database=my_company;Data Source=localhost;User Id=root;Password=pwd";

   MySqlConnection connection = new MySqlConnection(Connect);

   string command = "select department_no, department_name, department_manager from
                        department";

         MySqlDataAdapter da = new MySqlDataAdapter(command, connection);
         DataTable dt = new DataTable();
         da.Fill(dt);

         Departments.DataContext = dt;

         Departments.SelectedIndex = 0;

         textBox_HourlyRate.Text = "0.00";

         BitmapImage bm1 = new BitmapImage();
```

```
        bm1.BeginInit();
        bm1.UriSource = new Uri("e:/images/noname.jpg", UriKind.Relative);
        bm1.CacheOption = BitmapCacheOption.OnLoad;
        bm1.EndInit();
        image1.Source = bm1;

    }
```

Для того, чтобы при заполнении выпадающий список имел вид, представленный на рис.4, к элементу управления ComboBox необходимо применить форматирование

Рис. 4. Выпадающий список при выборе значений

```
<ComboBox Height="26" HorizontalAlignment="Left" Margin="35,226,0,0" Name="Departments" VerticalAlign-
ment="Top" Width="181" ItemsSource="{Binding}" SelectionChanged="Departments_SelectionChanged">
            <ComboBox.ItemTemplate>
          <DataTemplate>
            <StackPanel Orientation="Horizontal">
              <TextBlock Margin="2" Text="{Binding department_no}"/>

            </StackPanel>
          </DataTemplate>
        </ComboBox.ItemTemplate>
        <ComboBox.ItemContainerStyle>
          <Style TargetType="{x:Type ComboBoxItem}">
            <Setter Property="SnapsToDevicePixels" Value="True"/>
            <Setter Property="OverridesDefaultStyle" Value="True"/>
            <Setter Property="Template">
              <Setter.Value>
                <ControlTemplate TargetType="{x:Type ComboBoxItem}">
```

```
                    <Border Name="templateBorder" Padding="2" SnapsToDevicePixels="true">
                      <ContentPresenter>
                        <ContentPresenter.Content>
                          <Grid>
                            <Grid.ColumnDefinitions>
                              <ColumnDefinition/>
                              <ColumnDefinition/>
                              <ColumnDefinition/>
                            </Grid.ColumnDefinitions>
                            <TextBlock Margin="5" Grid.Column="0" Text="{Binding department_no}"/>
                            <TextBlock Margin="5" Grid.Column="1" Text="{Binding department_name}"/>
                            <TextBlock Margin="5" Grid.Column="2" Text="{Binding department_manager}"/>
                          </Grid>
                        </ContentPresenter.Content>
                      </ContentPresenter>
                    </Border>
                    <ControlTemplate.Triggers>
                      <Trigger Property="IsHighlighted" Value="True">
                        <Setter Property="Foreground" Value="{x:Static SystemColors.HighlightTextBrush}"/>
                        <Setter TargetName="templateBorder" Property="Background" Value="{x:Static SystemCo-
lors.HighlightBrush}"/>
                      </Trigger>
                    </ControlTemplate.Triggers>
                  </ControlTemplate>
                </Setter.Value>
              </Setter>
            </Style>
          </ComboBox.ItemContainerStyle>
        </ComboBox>
```

Событие **SelectionChanged** для обработки любого изменения ComboBox, вызванного пользователем или кодом будет иметь вид

```
private void Departments_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            Departments_Value();
        }

        private string Departments_Value()
        {
            DataRowView myDataRowView = Departments.SelectedItem as DataRowView;

            string sValue = string.Empty;
            if (myDataRowView != null)
            {
                sValue = oDataRowView.Row["department_no"] as string;
            }
            return sValue;
        }
```

Фотография может быть загружена из файла

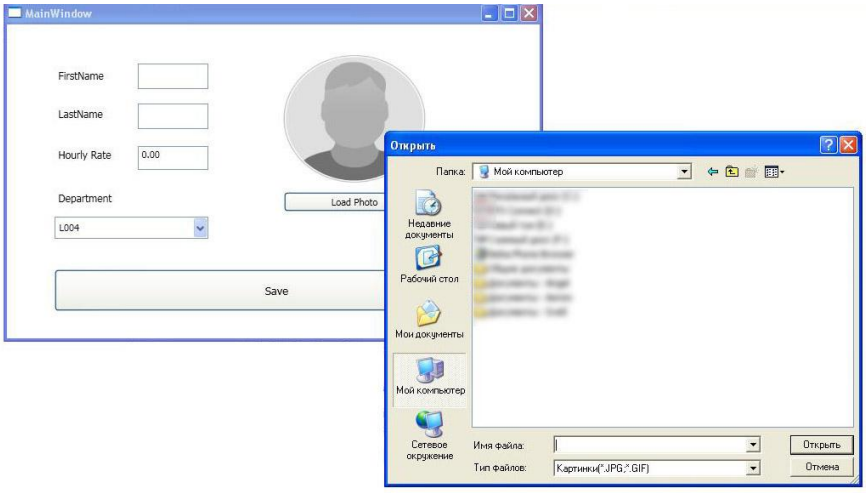


Рис. 5. Выбор файла с фотографией сотрудника

Событие кнопки LoadPhoto_Click нажатия кнопки, вызванное пользователем

```
private void LoadPhoto_Click(object sender, RoutedEventArgs e)
    {
       OpenFileDialog myDialog = new OpenFileDialog();
       myDialog.Filter = "Images(*.JPG;*.GIF;*.PNG)|*.JPG;*.GIF;*.PNG" + "|All files (*.*)|*.* ";
       myDialog.CheckFileExists = true;
       if (myDialog.ShowDialog() == true)
          {
          try
             {
                Photo = myDialog.FileName;
                BitmapImage bm1 = new BitmapImage();
                bm1.BeginInit();
                bm1.UriSource = new Uri(myDialog.FileName, UriKind.Relative);
                bm1.CacheOption = BitmapCacheOption.OnLoad;
                bm1.EndInit();
                image1.Source = bm1;
             }
             catch
             {
                MessageBox.Show("Unable to Load Image");
                image1.Source = new BitmapImage(new Uri("Images/noname1.png", Uri-
Kind.Relative));
             }
          }
       }
```

Событие кнопки Save_Click нажатия кнопки Save, вызванное пользователем, приведет к записи данных в таблицу

```
private void Save_Click(object sender, RoutedEventArgs e)
      {

    string Connect = "Database=my_company;Data Source=localhost;User Id=root;Password=pwd";

    MySqlConnection con = new MySqlConnection(Connect);
    try
         {
            con.Open();

            MySqlCommand comm = con.CreateCommand();

            string first_name, last_name;

// проверяем заполнение текстовых полей. Отсутствие значений приводит к вызову
// пользовательского диалогового окна ввода данных

            if (string.IsNullOrEmpty(textBox_FirstName.Text) ||
               string.IsNullOrWhiteSpace(textBox_FirstName.Text))
          {

            InputControl inputDialog = new InputControl("Please enter your name:", "");
               if (inputDialog.ShowDialog() == true)
                  textBox_FirstName.Text = inputDialog.Answer;
              }
      if (string.IsNullOrEmpty(textBox_LastName.Text) ||
         string.IsNullOrWhiteSpace(textBox_LastName.Text))
        {
         InputControl inputDialog = new InputControl("Please enter your family name:", "");
               if (inputDialog.ShowDialog() == true)
                  textBox_LastName.Text = inputDialog.Answer;
            }

// Пытаемся обезопасить ввод от SQL-injection
             if (!Regex.IsMatch(textBox_FirstName.Text, @"^[a-zA-Z]{1,20}$"))
            {       MessageBox.Show("First Name - Invalid text");
                  textBox_FirstName.Text=String.Empty;
                  first_name = String.Empty;
            }
            else {   first_name=textBox_FirstName.Text; };

            if (!Regex.IsMatch(textBox_LastName.Text, @"^[a-zA-Z]{1,20}$"))
            {   MessageBox.Show("Last Name - Invalid text");
               textBox_LastName.Text = String.Empty;
               last_name = String.Empty;
            }
```

```
            else   {     last_name = textBox_LastName.Text;        };

// получаем выбранное в ComboBox значение департамента

            string department_no = Departments_Value();

// устанавливаем разделить между целой и дробной частью числа

            string CultureName = Thread.CurrentThread.CurrentCulture.Name;
            CultureInfo ci = new CultureInfo(CultureName);
            if (ci.NumberFormat.NumberDecimalSeparator != ".")
            {
               ci.NumberFormat.NumberDecimalSeparator = ".";
               Thread.CurrentThread.CurrentCulture = ci;
            };

            double hourly_rate;

           if (!string.IsNullOrEmpty(textBox_FirstName.Text) ||
                    (!string.IsNullOrEmpty(textBox_LastName.Text)))
            {
              if (double.TryParse(textBox_HourlyRate.Text, out hourly_rate))
             {
               comm.CommandText = "INSERT INTO employee (first_name, last_name, hourly_rate,
                                    department_no) VALUES ('" + first_name + "','" + last_name
                                    + "'," + hourly_rate + ",'" + department_no + "');";
               MessageBox.Show(comm.CommandText);
               comm.ExecuteNonQuery();
             }
             else
             {
               MessageBox.Show("Missing hourly rate" );
             }
            }
             else
            {
               MessageBox.Show(" Missing record" );
            };


           if ((string.IsNullOrEmpty(textBox_FirstName.Text)) &&
                              (string.IsNullOrEmpty(textBox_LastName.Text)))
            {
               MessageBox.Show("Missing file name");
            }
            else
            {
               JpegBitmapEncoder jpegBitmapEncoder = new JpegBitmapEncoder();

             jpegBitmapEncoder.Frames.Add(BitmapFrame.Create(image1.Source as BitmapSource));
```

```
            FileStream fileStream = new FileStream(@"e:\\images\" + textBox_FirstName.Text +
                                                   textBox_LastName.Text + ".jpg", FileMode.Create);
            jpegBitmapEncoder.Save(fileStream);

            fileStream.Close();

            }
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }
```

Незаполнение текстовых полей приводит к вызову окна, код которого приведен ниже

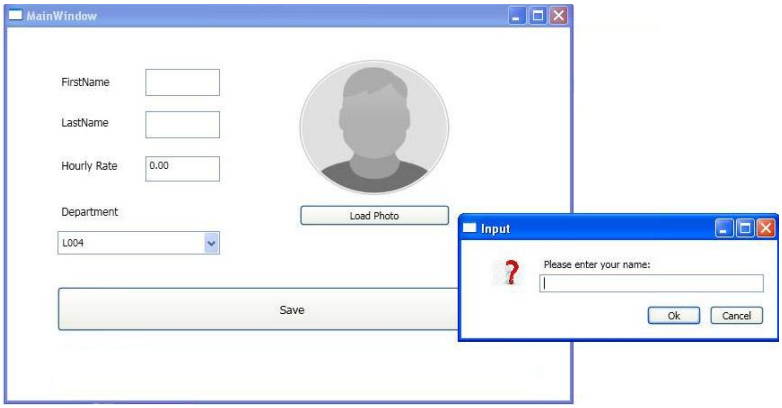


Рис. 6. Проверка заполнения окна формы

Окно InputControl реализовано в том же проекте WpfApplication

```
namespace WpfApplication
{
   public partial class InputControl : Window
   {
      public InputControl(string question, string defaultAnswer = "")
      {
         InitializeComponent();
         lblQuestion.Content = question;
         txtAnswer.Text = defaultAnswer;
      }

         private void btnDialogOk_Click(object sender, RoutedEventArgs e)
```

```
            {
                this.DialogResult = true;
            }
        private void Window_ContentRendered(object sender, EventArgs e)
            {
                txtAnswer.SelectAll();
                txtAnswer.Focus();
            }
        public string Answer
            {
                get { return txtAnswer.Text; }
            }
    }

}
```

Если фотографии сотрудников хранятся в базе данных как тип данных blob, то событие нажатие кнопки Save может быть реализовано следующим образом:

```
private void Save_Click(object sender, RoutedEventArgs e)
      {
  string Connect = "Database=my_company;Data Source=localhost;User Id=root;Password=pwd ";

  MySqlConnection con = new MySqlConnection(Connect);
  con.Open();
  MySqlCommand comm = con.CreateCommand();

 // Конвертируем рисунок image1.Source (фото сотрудника),
//  загруженное в элемент управления  image1 для последующей записи в базу

        byte[] ImageData;
        byte[] bytes = null;
        var bitmapSource = image1.Source as BitmapSource;

        if (bitmapSource != null)
        {
           var encoder = new JpegBitmapEncoder();
           encoder.Frames.Add(BitmapFrame.Create(bitmapSource));

           using (var stream = new MemoryStream())
           {
              encoder.Save(stream);
              bytes = stream.ToArray();
           }
        }
        ImageData = bytes;
```

```
 // параметризованный запрос является более защищенным от внедрения SQL-injection

        comm.CommandText = "INSERT INTO employee_photo (first_name, last_name,
                                    hourly_rate, department_no,photo) VALUES ( ?first_name,
                                   ?last_name, ?hourly_rate, ?department_no,?photo )";

// Предполагаем, что поля textBox_FirstName, textBox_FirstName, textBox_HourlyRate.Text
// заполнены. Может быть применена проверка заполнения полей
// как в предыдущем варианте метода Save_Click()

        comm.Parameters.AddWithValue("?first_name", textBox_FirstName.Text);
        comm.Parameters.AddWithValue("?last_name", textBox_FirstName.Text);

          string CultureName = Thread.CurrentThread.CurrentCulture.Name;
           CultureInfo ci = new CultureInfo(CultureName);
           if (ci.NumberFormat.NumberDecimalSeparator != ".")
           {
              ci.NumberFormat.NumberDecimalSeparator = ".";
              Thread.CurrentThread.CurrentCulture = ci;
           };

           double hourly_rate;

           if (double.TryParse(textBox_HourlyRate.Text, out hourly_rate))

           { comm.Parameters.Add("?hourly_rate", MySqlDbType.Decimal).Value = hourly_rate; }

           string department_no = Departments_Value();

           comm.Parameters.AddWithValue("department_no", department_no);

           comm.Parameters.Add("?photo", MySqlDbType.LongBlob).Value = ImageData;

        int RowsAffected = comm.ExecuteNonQuery();

        if (RowsAffected > 0)
        {
           MessageBox.Show("Image saved sucessfully!");
        }

        con.Close();         }
```

Изображение может быть загружено из файла, путь к которому определен при выборе файла (см выше код LoadPhoto_Click())

```
if (myDialog.ShowDialog() == true)   {try   { Photo = myDialog.FileName; …
```

Код записи в базу данных выбранного файла будет иметь вид:

```
FileStream fs;
BinaryReader br;
byte[] ImageData;
fs = new FileStream(Photo, FileMode.Open, FileAccess.Read);
br = new BinaryReader(fs);
ImageData = br.ReadBytes((int)fs.Length);
br.Close();
fs.Close();
comm.CommandText = "INSERT INTO employee_photo (photo) VALUES ('"
                        + ImageData + "')";
comm.ExecuteNonQuery();
```

Чтение информации из базы данных. Объединение таблиц

Используя визуальный редактор, создадим следующий интерфейс

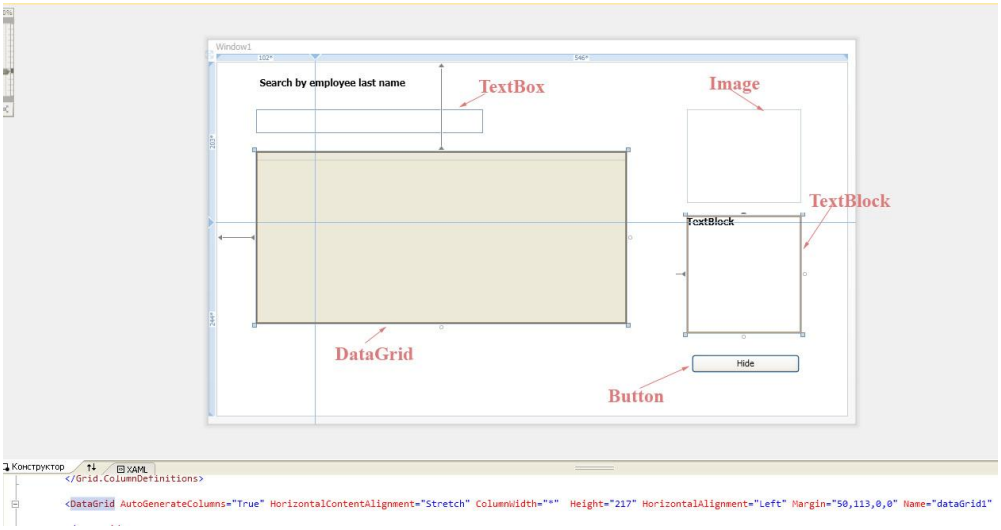


Рис. 7. Разметка окна формы вывода данных о сотрудниках

Информация о сотрудниках может быть загружена в элемент управления DataGrid. У элемента DataGrid установлено AutoGenerateColumns="True" и ширина столбцов ColumnWidth="*". Более подробная информация о сотруднике (фотография, персональные данные, данные об отделе и его руководителе) может быть представлена в правой части окна по щелчку мыши в соответствующей строке DataGrid.

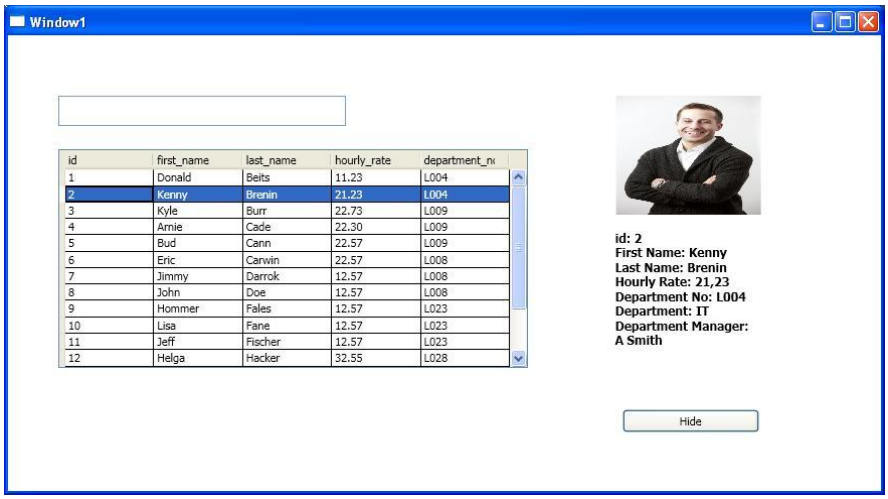


Рис. 8. Форма вывода данных о сотрудниках

Информация о сотруднике формируется по запросу к двум таблицам employee_photo и department.

```
using MySql.Data.MySqlClient;
using System.Data;
using MySql.Data;

namespace WpfApplication
{

   public partial class Window1 : Window
   {
      DataTable dt = new DataTable();

      public Window1()
      {
         InitializeComponent();
      }

    private void Window_Loaded(object sender, RoutedEventArgs e)
   {
    string Connect = "Database=my_company;Data Source=localhost;User Id=root;Password=pwd";

   string sql = "SELECT id,first_name, last_name, hourly_rate, department_no
                 FROM employee_photo ;";

// заполняем таблицу DataGrid данными из таблицы employee

  using (MySqlConnection connection = new MySqlConnection(Connect))
            {
```

```
                connection.Open();
                using (MySqlCommand cmd = new MySqlCommand(sql, connection))
                {

                    MySqlDataAdapter da = new MySqlDataAdapter(cmd);
                    da.Fill(dt);
                    dataGrid1.DataContext = dt;

                }
                connection.Close();

// скрываем отображение данных о выбранном сотруднике в правой части окна

                textBlock1.Visibility = Visibility.Hidden;
                textBlock1.Text = string.Empty;
                button1.Visibility = Visibility.Hidden;
            }
        }

// Событие textBox1_TextChanged позволит фильтровать таблицу

      private void textBox1_TextChanged(object sender, TextChangedEventArgs e)
      {
         DataView dv = dt.DefaultView; ;
         dv.RowFilter = string.Format("last_name like '{0}%'", textBox1.Text);

      }

// Событие dataGrid1_SelectedCellsChanged позволит отобразить в правой части окна данные
//  о сотруднике согласно сделанному выбору – клик мышкой по строке таблицы.
// Данные о сотруднике формируются на основании результата запроса к таблицам employee //
и department

 private void dataGrid1_SelectedCellsChanged(object sender, SelectedCellsChangedEventArgs e)
      {
         if (dataGrid1.SelectedIndex >= 0)
         {
  string Connect = "Database=my_company;Data Source=localhost;User Id=root;Password=pwd ";

            string x = dt.DefaultView[dataGrid1.SelectedIndex]["id"].ToString();

     string sql = "SELECT id, first_name, last_name, hourly_rate, department.department_no,
                     photo, department_name, department_manager " +
                    "FROM employee_photo inner join department on
                    employee_photo.department_no=department.department_no where id='" + x + "';";

            image1.Visibility = Visibility.Visible;
            button1.Visibility = Visibility.Visible;

            using (MySqlConnection connection = new MySqlConnection(Connect))
```

```
            {
                connection.Open();
                MySqlCommand command = new MySqlCommand(sql, connection);
                MySqlDataReader reader = command.ExecuteReader();

                if (reader.HasRows) // если есть данные
                {
                    while (reader.Read()) // построчно считываем данные
                    {

                        object id = reader.GetValue(0);
                        object first_name = reader.GetValue(1);
                        object last_name = reader.GetValue(2);
                        object hourly_rate = reader.GetValue(3);
                        object department_no = reader.GetValue(4);
                        object department_name = reader.GetValue(6);
                        object department_manager = reader.GetValue(7);

// считываем фотографию сотрудника из столбца photo типа blob таблицы employee_photo
            try
                {
                        byte[] data = (byte[])reader[5];

                        using (System.IO.MemoryStream ms = new System.IO.MemoryStream(data))
                        {
                            var imageSource = new BitmapImage();
                            imageSource.BeginInit();
                            imageSource.StreamSource = ms;
                            imageSource.CacheOption = BitmapCacheOption.OnLoad;
                            imageSource.EndInit();

                            image1.Source = imageSource;
                        }
                    }
        catch {
                MessageBox.Show("Missing Photo");
                image1.Source = new BitmapImage(new Uri("Images/noname.jpg", UriKind.Relative));
               };

               textBlock1.Text = "id: " + Convert.ToString(id) + "\n"
                                 + "First Name: " + Convert.ToString(first_name) + "\n"
                                 + "Last Name: " + Convert.ToString(last_name) + "\n"
                                 + "Hourly Rate: " + Convert.ToString(hourly_rate) + "\n"
                                 + "Department No: " + Convert.ToString(department_no) + "\n"
```

```
                                + "Department: " + Convert.ToString(department_name) + "\n"
                                + "Department Manager: " + "\n" +
                                   Convert.ToString(department_manager);
                        }
                    }
                reader.Close();
                    }
                                textBlock1.Visibility = Visibility.Visible;
                }
            }
            private void Hide_Click(object sender, RoutedEventArgs e)
            {
                button1.Visibility = Visibility.Hidden;
                image1.Visibility = Visibility.Hidden;
                textBlock1.Visibility = Visibility.Hidden;
            }
        }
    }
```

Данные в элементе управления DataGrid могут быть отфильтрованы по значению поля, что позволяет быстро найти информацию о сотрудниках.

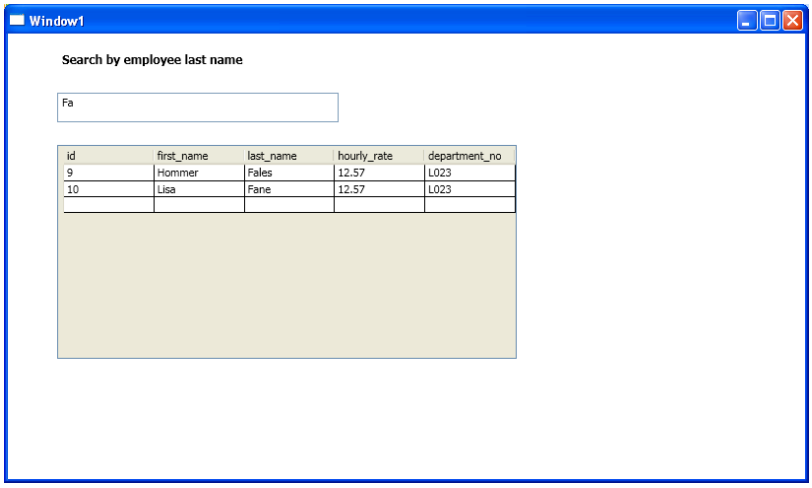


Рис. 9. Фильтрация данных в элементе управления DataGrid

Удаление записей из базы данных

Записи из базы данных могут быть удалены. Окно, позволяющее удалить данные о сотрудниках, может иметь вид:

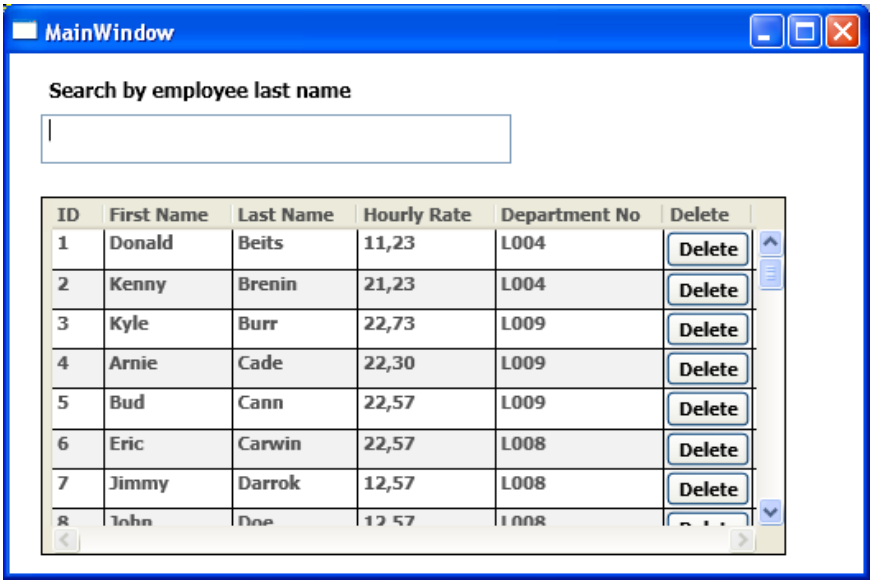


Рис. 10. Окно для удаления данных о сотрудниках

В элементе управления DataGrid добавлен столбец с кнопками Delete. Событие нажатия на кнопку связано с btnDelete_Click(). DataGrid переименована в "Employees". Разметка приведена ниже.

```
<DataGrid Name="Employees" AutoGenerateColumns="False" CanUserAddRows="False"
Margin="5,5,220,12" HorizontalAlignment="Stretch" VerticalAlignment="Stretch"
RowBackground="#fff" FontWeight="Bold" Foreground="#525252"
ScrollViewer.CanContentScroll="True" MaxHeight="390"
AlternatingRowBackground="#f2f2f2" BorderBrush="#000" BorderThickness="1"
ScrollViewer.HorizontalScrollBarVisibility="Visible"
ScrollViewer.VerticalScrollBarVisibility="Auto">
          <DataGrid.Columns>
               <DataGridTextColumn Header="first_Name" Binding="{Binding Path='First_Name'}"
IsReadOnly="True" />
               <DataGridTextColumn Header="Last_Name" Binding="{Binding Path='Last_Name'}"
IsReadOnly="True" />
               <DataGridTextColumn Header="Email" Binding="{Binding Path='Email'}" IsReadOn-
ly="True" />

               <DataGridTemplateColumn Header="Delete" >
                   <DataGridTemplateColumn.CellTemplate>
                       <DataTemplate>
                        <Button Name="btnDelete" Content="Delete" Click="btnDelete_Click" />
                       </DataTemplate>
                   </DataGridTemplateColumn.CellTemplate>
               </DataGridTemplateColumn>
           </DataGrid.Columns>
       </DataGrid>
```

Удаление информации о сотрудниках может быть реализовано следующим образом.

```
using MySql.Data.MySqlClient;
using System.Data;
using MySql.Data;

namespace WpfApplication1
{
      public partial class MainWindow : Window
   {
      DataTable dt = new DataTable();

      public MainWindow()
      {
         InitializeComponent();
      }

    private void Window_Loaded(object sender, RoutedEventArgs e)
  {
   string Connect = "Database=my_company;Data Source=localhost;User Id=root;Password=pwd";

         string sql = "SELECT * FROM employee";

          using (MySqlConnection connection = new MySqlConnection(Connect)){

            connection.Open();
            MySqlCommand command = new MySqlCommand(sql, connection);
            MySqlDataReader reader = command.ExecuteReader();

            if (reader.HasRows) // если есть данные
            {
               dt.Columns.Add("ID", typeof(String));
               dt.Columns.Add("First_Name", typeof(String));
               dt.Columns.Add("Last_Name", typeof(String));
               dt.Columns.Add("Hourly_Rate", typeof(String));
               dt.Columns.Add("Department_No", typeof(String));

              while (reader.Read()) // построчно считываем данные
               {
                  object id = reader.GetValue(0);
                  object first_name = reader.GetValue(1);
                  object last_name = reader.GetValue(2);
                  object hourly_rate = reader.GetValue(3);
                  object department_no = reader.GetValue(4);

                  dt.Rows.Add(id.ToString(), first_name.ToString(), last_name.ToString(),
                              hourly_rate.ToString(), department_no.ToString());
               }

               Employees.ItemsSource = dt.DefaultView;
```

```
            }

     reader.Close();
      }
  }

     private void btnDelete_Click(object sender, RoutedEventArgs e)
     {
        try
        {
           DataTable dt1 = new DataTable();

           dt1 = dt;
           int index = Employees.SelectedIndex;
           int id = Convert.ToInt32(dt1.DefaultView[index]["id"]);
           DataRow b = dt1.Rows[index];
           dt1.Rows.Remove(b);
           Employees.ItemsSource = dt1.DefaultView;

   string Connect = "Database=my_company;Data Source=localhost;User Id=root;Password=pwd";

            string sql = "delete FROM employee where id="+id+";";

           MySqlConnection connection = new MySqlConnection(Connect);
           connection.Open();
           MySqlCommand command = new MySqlCommand(sql, connection);

           int countLine = command.ExecuteNonQuery();

           MessageBox.Show("Количество удаленных строк "+ countLine);

           connection.Close();

        }
        catch (Exception ex)
        {
           MessageBox.Show(ex.Message.ToString());
        }
     }
 private void textBox1_TextChanged(object sender, TextChangedEventArgs e)
     {
        DataView dv = dt.DefaultView;

        dv.RowFilter = string.Format("Last_Name like '{0}%'", textBox1.Text);

      }
  }
}
```

Обобщение данных с помощью агрегирующих функций

При статистическом анализе баз данных необходимо получать такую информацию, как общее количество записей, наибольшее и наименьшее значения заданного поля записи, усредненное значение поля и т. д. Данная задача выполняется с помощью запросов, содержащих так называемые агрегирующие функции. Создадим приложение, позволяющее получить информацию о средней, максимальной и минимальной оплате за час, а также количестве сотрудниклв некоторого отдела. Приложение может иметь вид:

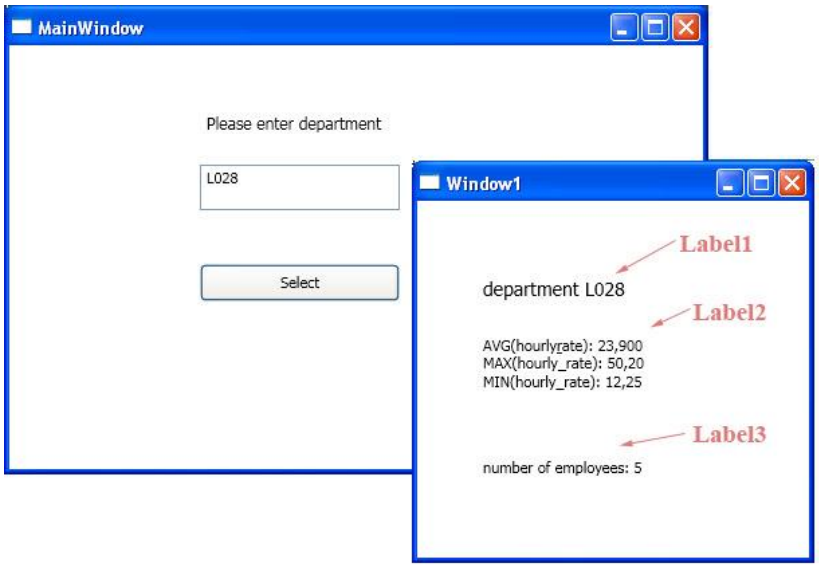


Рис. 11. Приложение для вычисления агрегирующих значений

```
namespace WpfApplication1
{

  public partial class MainWindow : Window
  {
    public MainWindow()
    {
      InitializeComponent();
    }

    private void button1_Click(object sender, RoutedEventArgs e)
```

```
    {
        Window1 myWindow = new Window1();
        myWindow.department_no = textBox1.Text;
        myWindow.Show();
    }
  }
}
```

При реализации окна с результатами агрегированных данных используются методы класса MySqlCommand: ExecuteScalar() и ExecuteReader(). Метод ExecuteScalar() выполняет запрос и возвращает значение первого поля первой строки из набора строк, сгенерированного командой. Этот метод обычно применяется при выполнении агрегатной команды select, использующей функции count(), sum(), avg() и т.д., для вычисления единственного значения. Метод ExecuteReader() выполняет запрос и возвращает объект DataReader, который является оболочкой однонаправленного курсора, доступного только для чтения. Метод ExecuteNonQuery() выполняет команды, такие как SQL-операторы вставки, удаления или обновления записей. Возвращаемое значение указывает количество строк, обработанных командой. Метод ExecuteNonQuery() также можно использовать для выполнения команд определения данных, которые создают, изменяют и уничтожают объекты базы данных (наподобие таблиц, индексов, ограничений и т.п.)

```
namespace WpfApplication1
{
  public partial class Window1 : Window
  {

// модификатор доступа public переменной  department_no открывает доступ
// к ней из класса MainWindow
    public string department_no = string.Empty;

    public Window1()
    {
      InitializeComponent();
    }
    private void Window_Loaded(object sender, RoutedEventArgs e)
    {
      label1.Content = "department " + department_no  ;

  string Connect = "Database=my_company;Data Source=localhost;User Id=root;Password=pwd";
```

```
        string sql1 = "SELECT AVG(hourly_rate) from employee where department_no='"
                      + department_no + "';";
        string sql2 = "SELECT MAX(hourly_rate) from employee where department_no='"
                      + department_no + "';";
        string sql3 = "SELECT MIN(hourly_rate) from employee where department_no='"
                      +department_no+"';";

        MySqlConnection connection = new MySqlConnection(Connect);

        connection.Open();

        MySqlCommand command = new MySqlCommand(sql1+sql2+sql3 , connection);

        MySqlDataReader reader = command.ExecuteReader();

       label2.Content = string.Empty;

        do{
            while (reader.Read())
            {
               for (int field = 0; field < reader.FieldCount; field++)
              {
                  label2.Content = label2.Content + reader.GetName(field).ToString() +
                                                  ": " + reader.GetValue(field).ToString() + "\n";
              }
            }
          } while (reader.NextResult());

//  используем reader.NextResult(),т.к.  команда, которая сгенерировала DataReader,
//  возвратила более одного набора строк – результаты запросов sql1, sql2, sql3

            reader.Close();

            string sql = "SELECT COUNT(id) from employee where department_no='"
                        + department_no + "';";

            command = new MySqlCommand(sql, connection);

          int i = Convert.ToInt32(command.ExecuteScalar());;

          label3.Content = "number of employees: " + i;

        connection.Close();
      }
   }
}
```

Перемещение по записям. Подзапросы

Рассмотрим распространенную задачу перемещения к последующей и предыдущей записи в выборке. Например, у нас есть запрос, результаты которого есть данные о сотрудниках некоторого департамента, например, L004. Тогда задача может быть решена следующим образом.

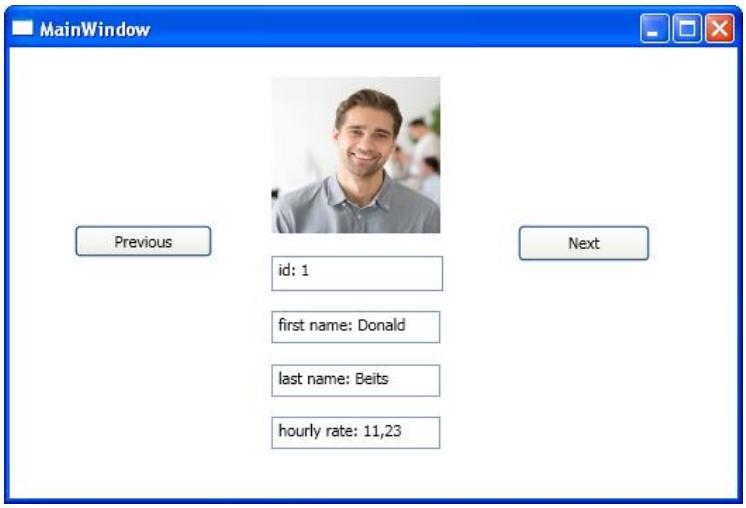


Рис. 12. Окно управления перемещением по записям

```
namespace WpfApplication1
{
    public partial class MainWindow : Window
  {
    int Record;

    public MainWindow()
    {
       InitializeComponent();
    }


 public void myRecord(string sql)
 {

  string Connect = "Database=my_company;Data Source=localhost;User Id=root;Password=pwd";

        using (MySqlConnection connection = new MySqlConnection(Connect))
        {
          connection.Open();
          MySqlCommand command = new MySqlCommand(sql, connection);
```

```
        MySqlDataReader reader = command.ExecuteReader();

        if (reader.HasRows)     {

            while (reader.Read())       {

                object id = reader.GetValue(0);
                object first_name = reader.GetValue(1);
                object last_name = reader.GetValue(2);
                object hourly_rate = reader.GetValue(3);

                Record = Convert.ToInt32(id);
                try
                {

                    BitmapImage bm1 = new BitmapImage();
                    bm1.BeginInit();
                    bm1.UriSource = new Uri(@"e:\\images\"+id+".jpg", UriKind.Relative);
                    bm1.CacheOption = BitmapCacheOption.OnLoad;
                    bm1.EndInit();
                    image1.Source = bm1;
                }
                catch
                {
                    MessageBox.Show("Missing Photo ");

                };

                textBox1.Text = "id: " + Convert.ToString(id);
                textBox2.Text = "first name: " + Convert.ToString(first_name);
                textBox3.Text = "last name: " + Convert.ToString(last_name);
                textBox4.Text = "hourly rate: " + Convert.ToString(hourly_rate);
            }
        }

        reader.Close();
    }
}


private void Window_Loaded(object sender, RoutedEventArgs e)
{

        string sql = "SELECT id, first_name, last_name, hourly_rate from employee where id=" +
                     "(select min(id) from employee_photo where department_no='L004');";

    myRecord(sql);

}
```

```
    private void NextButton_Click(object sender, RoutedEventArgs e)
    {
        string sql = "SELECT id, first_name, last_name, hourly_rate from employee where"+
                     " (department_no='L004' AND id>" + Record + " ) ORDER BY id limit 1;";

        myRecord(sql);

    }

    private void PreviousButton_Click(object sender, RoutedEventArgs e)
    {
        string sql = "SELECT id, first_name, last_name, hourly_rate from employee where"+
                  "(department_no='L004' AND id<" + Record + " ) ORDER BY id DESC limit 1;";

        myRecord(sql);
    }
  }
}
```

Контрольные вопросы

1. Интерфейсы программирования приложений для доступа к базам данных.
2. Сформулировать SQL-запрос, позволяющий найти сотрудников, имеющих максимальную и вторую по величине зарплату.
3. Сформулировать SQL-запрос, позволяющий найти максимальную зарплату сотрудников каждого отдела.
4. Сформулировать SQL-запрос, позволяющий определить отделы, в которых число сотрудников не превышает 20.
5. Сформулировать SQL-запрос, позволяющий вывести полную информацию о сотруднике, включая название департамента.
6. Сформулировать SQL-запрос, позволяющий найти сотрудника, чья оплата за час превышает 200.
7. Ввести в таблицу сотрудники столбец со значениями фактически отработанного сотрудником времени.
8. Сформулировать SQL-запрос, позволяющий вычислить зарплату сотрудника.
9. Сформулировать SQL-запрос, позволяющий найти суммарную заработную плату каждого отдела.
10. Сформулировать SQL-запрос, позволяющий начислить премию сотруднику по заданному алгоритму.
11. Сформулировать SQL-запрос, позволяющий найти имя сотрудника, чье имя начинается с ‘M’.
12. Сформулировать SQL-запрос, позволяющий найти все записи о сотрудниках, содержащие заданное слово.
13. Сформулировать SQL-запрос для поиска дубликатов строк в базе данных.
14. Написать SQL-запрос удаляющий дубликаты строк в базе данных.
15. Сформулировать SQL-запрос, позволяющий найти всех сотрудников, чья оплата за час выше средней.
16. Сформулировать SQL-запрос, позволяющий удалить записи сотрудников при расформировании отдела.
17. Сформулировать SQL-запрос, позволяющий найти количество сотрудников в каждом отделе.
18. Написать триггер, сохраняющий при увольнении данные о сотруднике в отдельную таблицу.
19. Сформулировать SQL-запрос, позволяющий найти всех подчиненных определенного менеджера.

Список литературы

1. Дейт К. Дж. Введение в системы баз данных. –СПб.: Изд. дом «Вильямс», 1999.– 848 с.
2. Коннолли Т. Базы данных: проектирование, реализация и сопровождение. Теория и практика.– М.: Изд. дом «Вильямс», 2000.– 1120 с.
3. Грабер М. Введение в SQL/M.– М.: Лори, 2010.– 228 с.
4. Дунаев В. В. Базы данных. Язык SQL.– СПб.:БХВ-Петербург, 2006.– 288с.
5. Андерсон К. Основы Windows Presentation Foundation. – СПб.: БХВ-Петербург, 2014– 432 с.
6. Мак-Дональд М. "WPF: Windows Presentation Foundation в .NET 4.5 с примерами на C# 5.0 для профессионалов". – М.: ООО "И.Д. Вильямс". 2013. – 629 с.
7. Троелсен, Эндрю. Язык программирования C# 5.0 и платформа .NET 4.5. – М.: ООО "И.Д. Вильямс", 2015. – 1392 с.
8. Натан А.WPF 4. Подробное руководство. – СПб.: БХВ-Петербург, 2011– 608 с.
9. Нейгел К., Ивьен Б., Глинн Дж., Уотсон К. C# 5.0 и платформа .NET 4.5 для профессионалов. М.: ООО "И.Д. Вильямс", 2015, 946 с.
10. Петцольд Ч., Microsoft Windows Presentation Foundation. Базовый курс. – М.: ООО "И.Д. Вильямс", 2014. – 944 с.

Светлана Павловна Никитенкова

РАЗРАБОТКА WPF-ПРИЛОЖЕНИЙ НА ОСНОВЕ БАЗ ДАННЫХ

Учебно-методическое пособие

Компьютерная верстка – С.П. Никитенкова

Федеральное государственное автономное образовательное учреждение
высшего образования «Национальный исследовательский
Нижегородский государственный университет им. Н.И. Лобачевского»

603950, Нижний Новгород, пр. Гагарина, 23