|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ | | | | | | | | | | |
|  | | | |  | | |  | | | |
| ФГБОУ ВПО «Пермский государственный национальный исследовательский университет» | | | | | | | | | | |
|  | |  | | | | | | |  | |
|  | | ОТЧЕТ  по лабораторной работе «Работа с данными в табличном представлении» по дисциплине «Учебная практика по базам данных» | | | | | | |  | |
|  | | |  | | | | |  | | |
|  | Работу выполнил  студент гр. ПМИ-1,2  Васильевых М.Г. \_\_\_\_\_\_\_\_\_\_  (подпись)  «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2021 | | | |  | Проверил  ассистент кафедры МОВС  Турова И.А. \_\_\_\_\_\_\_\_\_\_  (подпись)  «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2021 | | | |  |
|  | |  | | | | | | |  | |
|  | | Пермь 2021 | | | | | | |  | |

Содержание

[1 Постановка задачи 3](#_Toc83312391)

[2 Описание предметной области и базы данных 4](#_Toc83312392)

[3 Описание функционала разработанной программы 5](#_Toc83312393)

[3.1. Внешний вид и функционал приложения 5](#_Toc83312394)

[3.2 Общий алгоритм решения задачи 7](#_Toc83312395)

[4 Реализация программы 8](#_Toc83312396)

[5 Тестирование программы 13](#_Toc83312397)

# 1 Постановка задачи

Написать программу, которая на графическом интерфейсе пользователя позволяет просматривать, добавлять, удалять, изменять записи в двух связанных таблицах из спроектированной ранее базы данных предметной области без необходимости задания и

отображения пользователю искусственных ключей. Пользователь должен иметь возможность просматривать, добавлять, удалять, изменять записи непосредственно в табличном представлении.

Для каждой из таблиц на пользовательском интерфейсе должно отображаться не менее 3 столбцов, причём для некоторых столбцов по крайней мере 1 таблицы должен использоваться элемент управления, отличный от DataGridViewTextBoxColumn (например, ComboBox / CheckBox / DateTimePicker / …).

# 2 Описание предметной области и базы данных

Для работы приложения нам таблицы «Поставщики» и «Банки» из базы данных «Поставки» - рисунок 1.1

В таблице «Поставщики» имеются записи:

id – идентификатор для поставщика, целочисленное значение, является ключом

name\_pr – название поставщика, символьное значение

id\_bank – идентификатор банка услугами которого пользуется поставщик, целочисленное значение

code\_payment – номер счёта поставщика, текстовое значение не имеющее на себе каких либо ограничений

count\_delays – кол- во раз когда поставщик опоздал с поставкой, целочисленное значение

data\_coop – дата когда был заключён контракт на поставки, тип поля дата

В таблице «Банк» имеются записи:

id – идентификатор банка, целочисленное значение, является ключом

name\_bank – название банка

lvl\_licenses – уровень лицензии банка, значение от 0 до 10, целочисленное

active – активы банка, неотрицательное целочисленное значение

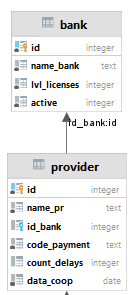


Рисунок 1.1 – Используемая БД

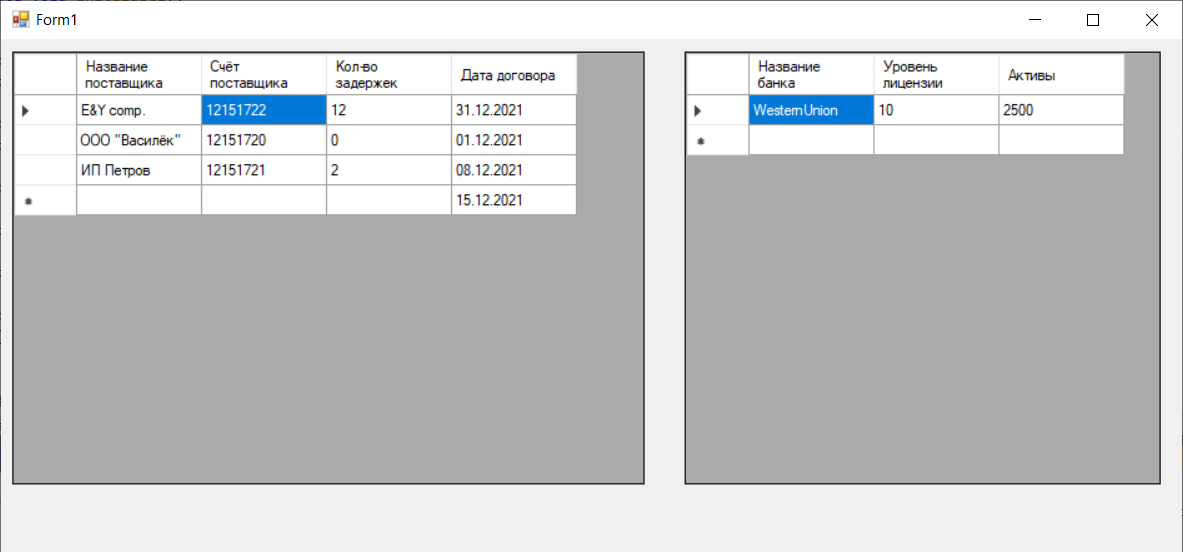
# 3 Описание функционала разработанной программы

3.1. Внешний вид и функционал приложения

При запуске приложения отображается главная форма. На ней расположены элементы управления: DataGridView (для отображения записей и взаимодействия с ними), в количестве 2 элементов.

1 для отображения записей таблицы «Поставщики», 2 для отображения связанной с ним таблицы «Банки».

Внешний вид приложения представлен на рис. 1.

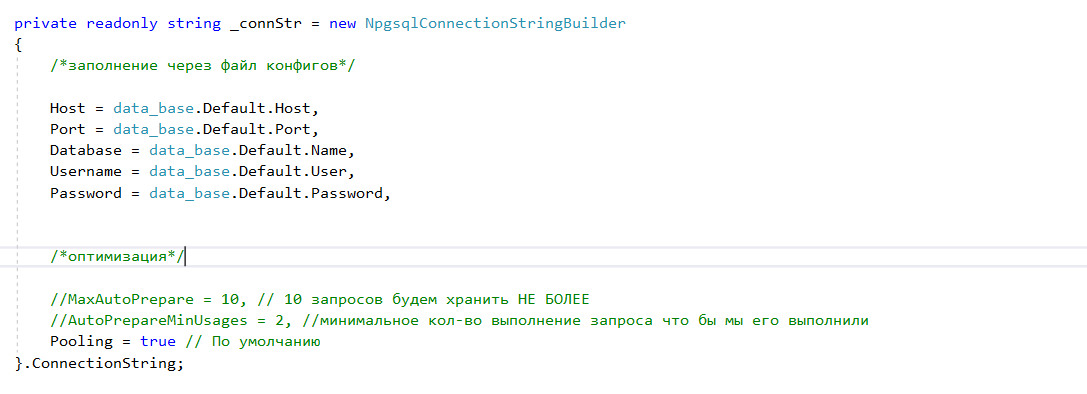


## 3.2 Общий алгоритм решения задачи

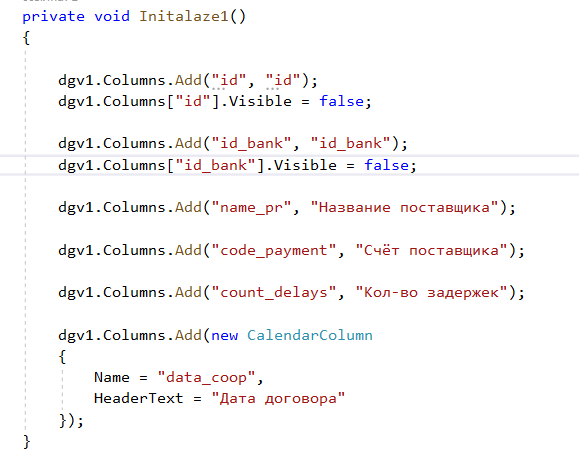
Данная задача подразумевает отображение, удаление и изменение записей исходных таблиц

1. Отображение: Для начала нам нужно создать запрос в БД на получение данных из 1 таблицы, где мы отобразим все поля кроме ключей. Так же нам необходимо понять какие именно записи банков мы должны выводить, я остановился на варианте, что мы выводим банка выделенной строки в 1 элементе. По условию нам сказано не выводить пользователю искусственные ключи, мы будем их хранить в элементе только в сокрытом виде.
2. Добавление, изменение: Так как подразумевается, что элемент DataGridView обрабатывает только изменение ячейки, то будем вводить маркер для определения того, что именно мы собираемся делать. А именно если в обрабатываемой строке есть id, то будем считать, что мы изменяем поле, иначе мы его добавляем. И в соответствии с этими правилами будем запускать тот или иной запрос, в соответствии с правилами, установленными в БД. Если же мы передумали и отказались от изменений в ячейке, то нам необходимо ввести обработку события для отмены изменений на форме и создания запроса в БД.
3. Удаление: Так как элемент имеет событие для удаления строки из таблицы, мы можем, опустит рассмотрения удаления из самой формы, упомянув, что нам необходимо просто отправить запрос в БД на удаление записи.

# 4 Реализация программы

Создание строки подключения, происходит при помощи файла настроек

Метод инициализации для первого элемента



Так как мы не можем отображать ключи столбец «id» не отображается, столбец отвечающий за дату имеет свой собственный тип

**public** **class** **CalendarColumn** : DataGridViewColumn

{

**public** **CalendarColumn**()

: **base**(**new** CalendarCell())

{

}

**public** **override** DataGridViewCell CellTemplate

{

**get**

{

**return** **base**.CellTemplate;

}

**set**

{

// Ensure that the cell used for the template is a CalendarCell.

**if** (**value** != **null** &&

!**value**.GetType().IsAssignableFrom(**typeof**(CalendarCell)))

{

**throw** **new** **InvalidCastException**("Must be a CalendarCell");

}

**base**.CellTemplate = **value**;

}

}

}

**public** **class** **CalendarCell** : DataGridViewTextBoxCell

{

**public** **CalendarCell**()

: **base**()

{

// Use the short date format.

**this**.Style.Format = "d";

}

**public** **override** **void** **InitializeEditingControl**(**int** rowIndex, **object**

initialFormattedValue, DataGridViewCellStyle dataGridViewCellStyle)

{

// Set the value of the editing control to the current cell value.

**base**.InitializeEditingControl(rowIndex, initialFormattedValue,

dataGridViewCellStyle);

CalendarEditingControl ctl =

DataGridView.EditingControl **as** CalendarEditingControl;

// Use the default row value when Value property is null.

**if** (**this**.Value == **null**)

{

ctl.Value = (DateTime)**this**.DefaultNewRowValue;

}

**else**

{

ctl.Value = (DateTime)**this**.Value;

}

}

**public** **override** Type EditType

{

**get**

{

// Return the type of the editing control that CalendarCell uses.

**return** **typeof**(CalendarEditingControl);

}

}

**public** **override** Type ValueType

{

**get**

{

// Return the type of the value that CalendarCell contains.

**return** **typeof**(DateTime);

}

}

**public** **override** **object** DefaultNewRowValue

{

**get**

{

// Use the current date and time as the default value.

**return** DateTime.Now;

}

}

}

**class** **CalendarEditingControl** : DateTimePicker, IDataGridViewEditingControl

{

DataGridView dataGridView;

**private** **bool** valueChanged = **false**;

**int** rowIndex;

**public** **CalendarEditingControl**()

{

**this**.Format = DateTimePickerFormat.Short;

}

// Implements the IDataGridViewEditingControl.EditingControlFormattedValue

// property.

**public** **object** EditingControlFormattedValue

{

**get**

{

**return** **this**.Value.ToShortDateString();

}

**set**

{

**if** (**value** **is** String)

{

**try**

{

// This will throw an exception of the string is

// null, empty, or not in the format of a date.

**this**.Value = DateTime.Parse((String)**value**);

}

**catch**

{

// In the case of an exception, just use the

// default value so we're not left with a null

// value.

**this**.Value = DateTime.Now;

}

}

}

}

// Implements the

// IDataGridViewEditingControl.GetEditingControlFormattedValue method.

**public** **object** **GetEditingControlFormattedValue**(

DataGridViewDataErrorContexts context)

{

**return** EditingControlFormattedValue;

}

// Implements the

// IDataGridViewEditingControl.ApplyCellStyleToEditingControl method.

**public** **void** **ApplyCellStyleToEditingControl**(

DataGridViewCellStyle dataGridViewCellStyle)

{

**this**.Font = dataGridViewCellStyle.Font;

**this**.CalendarForeColor = dataGridViewCellStyle.ForeColor;

**this**.CalendarMonthBackground = dataGridViewCellStyle.BackColor;

}

// Implements the IDataGridViewEditingControl.EditingControlRowIndex

// property.

**public** **int** EditingControlRowIndex

{

**get**

{

**return** rowIndex;

}

**set**

{

rowIndex = **value**;

}

}

// Implements the IDataGridViewEditingControl.EditingControlWantsInputKey

// method.

**public** **bool** **EditingControlWantsInputKey**(

Keys key, **bool** dataGridViewWantsInputKey)

{

// Let the DateTimePicker handle the keys listed.

**switch** (key & Keys.KeyCode)

{

**case** Keys.Left:

**case** Keys.Up:

**case** Keys.Down:

**case** Keys.Right:

**case** Keys.Home:

**case** Keys.End:

**case** Keys.PageDown:

**case** Keys.PageUp:

**return** **true**;

**default**:

**return** !dataGridViewWantsInputKey;

}

}

// Implements the IDataGridViewEditingControl.PrepareEditingControlForEdit

// method.

**public** **void** **PrepareEditingControlForEdit**(**bool** selectAll)

{

// No preparation needs to be done.

}

// Implements the IDataGridViewEditingControl

// .RepositionEditingControlOnValueChange property.

**public** **bool** RepositionEditingControlOnValueChange

{

**get**

{

**return** **false**;

}

}

// Implements the IDataGridViewEditingControl

// .EditingControlDataGridView property.

**public** DataGridView EditingControlDataGridView

{

**get**

{

**return** dataGridView;

}

**set**

{

dataGridView = **value**;

}

}

// Implements the IDataGridViewEditingControl

// .EditingControlValueChanged property.

**public** **bool** EditingControlValueChanged

{

**get**

{

**return** valueChanged;

}

**set**

{

valueChanged = **value**;

}

}

// Implements the IDataGridViewEditingControl

// .EditingPanelCursor property.

**public** Cursor EditingPanelCursor

{

**get**

{

**return** **base**.Cursor;

}

}

**protected** **override** **void** **OnValueChanged**(EventArgs eventargs)

{

// Notify the DataGridView that the contents of the cell

// have changed.

valueChanged = **true**;

**this**.EditingControlDataGridView.NotifyCurrentCellDirty(**true**);

**base**.OnValueChanged(eventargs);

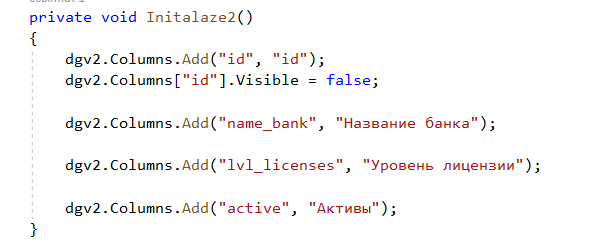
}

}



Метод для отображения данных на 1 элементе, так как при изменении ячейки мы можем откатить изменения, то нам необходимо хранить предыдущие значения в поле. Для этого в тэге строки мы храним словарь, где ключ это название столбца, а в поле значения, значение в ячейке.

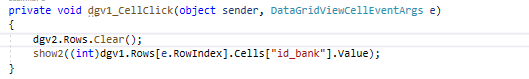
Аналогична структура 2 элемента.



И отображения данных на элементе.

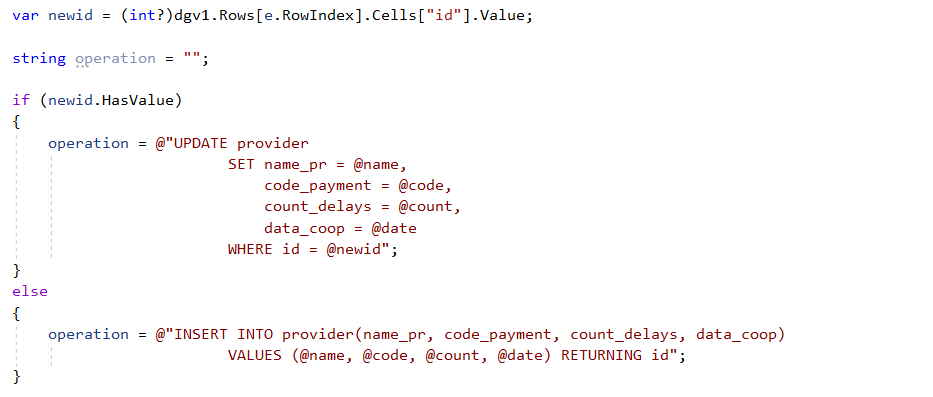


Так как нам необходимо выводит значения во 2 элемент соответственно выбранному 1 элементу, нам необходимо привязать событие выделения строки в 1 элементе и реакции на это 2 элемента.

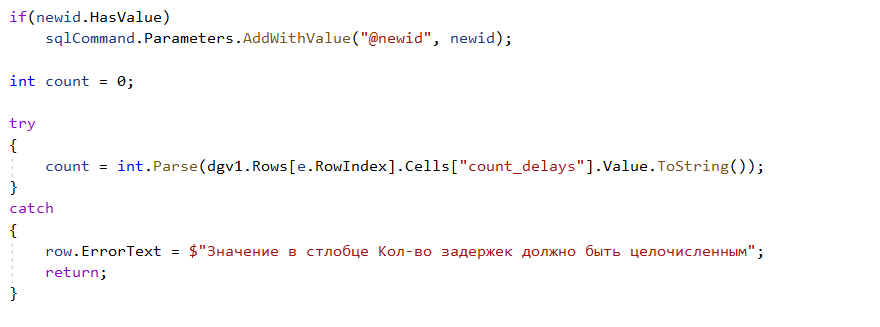


Обработка события изменения и добавления полей в 1 элементе

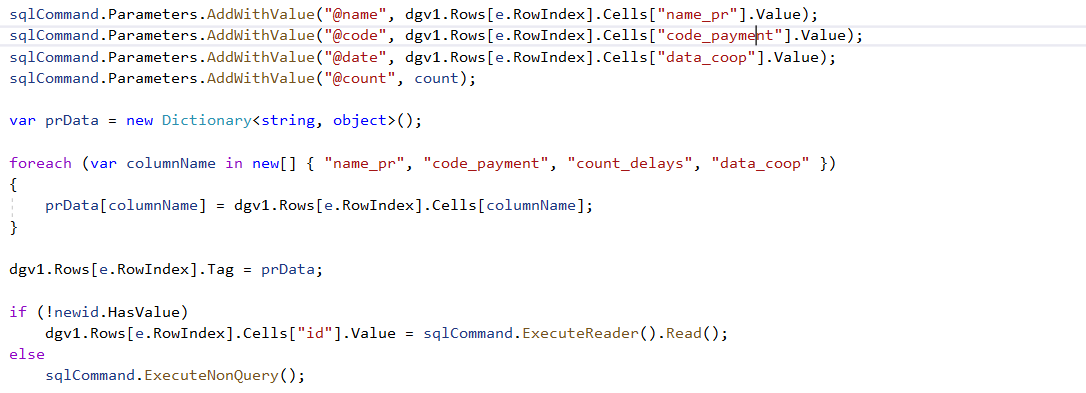
Проверка столбцов на пустоту

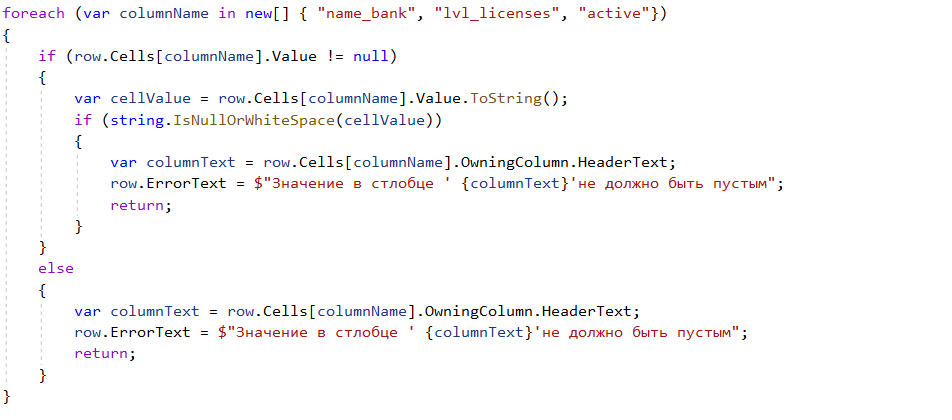
Формирование запроса в зависимости от контекста, а именно если в выбранной строке существует id, то вызываем запрос изменения, если id, то нет значит добавления.

Запрос добавления возвращает id новой записи в БД так нам необходимо его внести в поле обрабатываемой строки для дальней работай с записью.

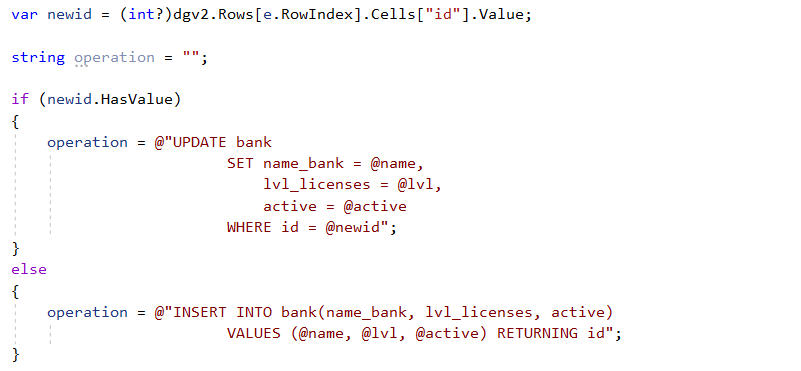
Так же накладывается ограничения из-за разных входных параметров для запросов.

Проверка значений.

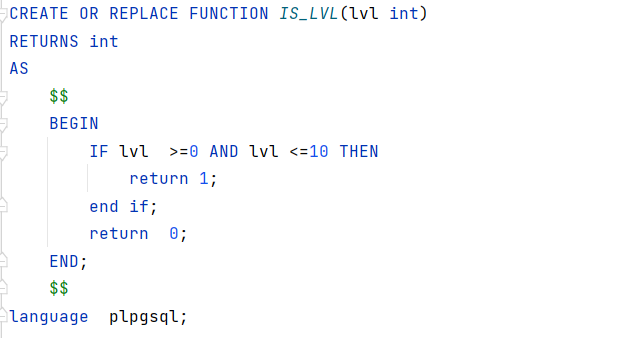
Передача параметров в запрос и заполнение тега записи заново, так как мы можем, как добавить новую запись, так и изменить её содержание.

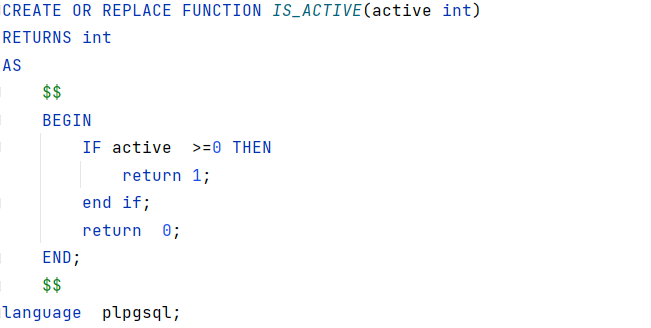
Проверка на пустоту записей при модификации и добавление полей для 2 элемента.

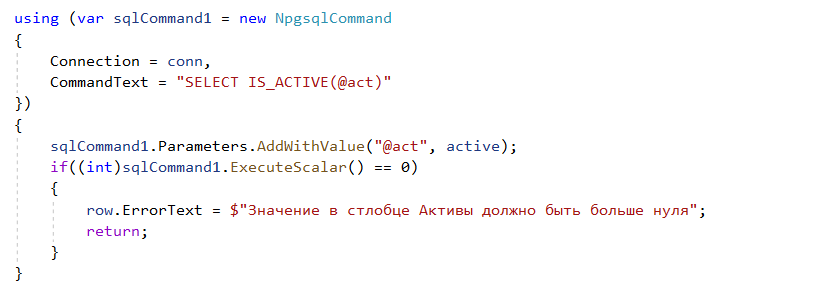
Аналогичное формирование запроса

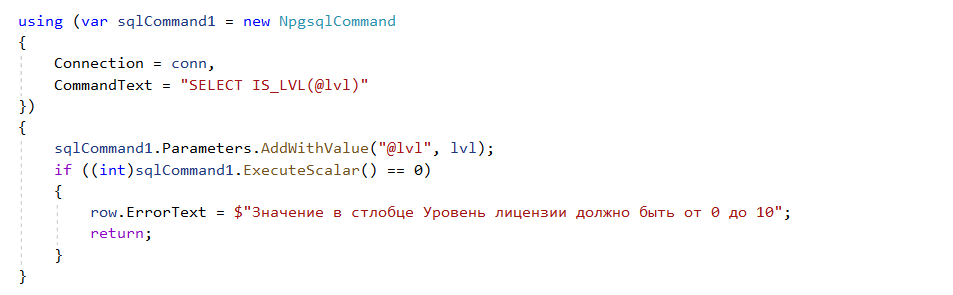
Аналогичное формирование запроса.

Так как на поля активов и уровня лицензии наложено логическое ограничение нам необходимо это проверять. Правильней всего это будет делать программными средствами базы данных. Для этого я написал 2 ХП.



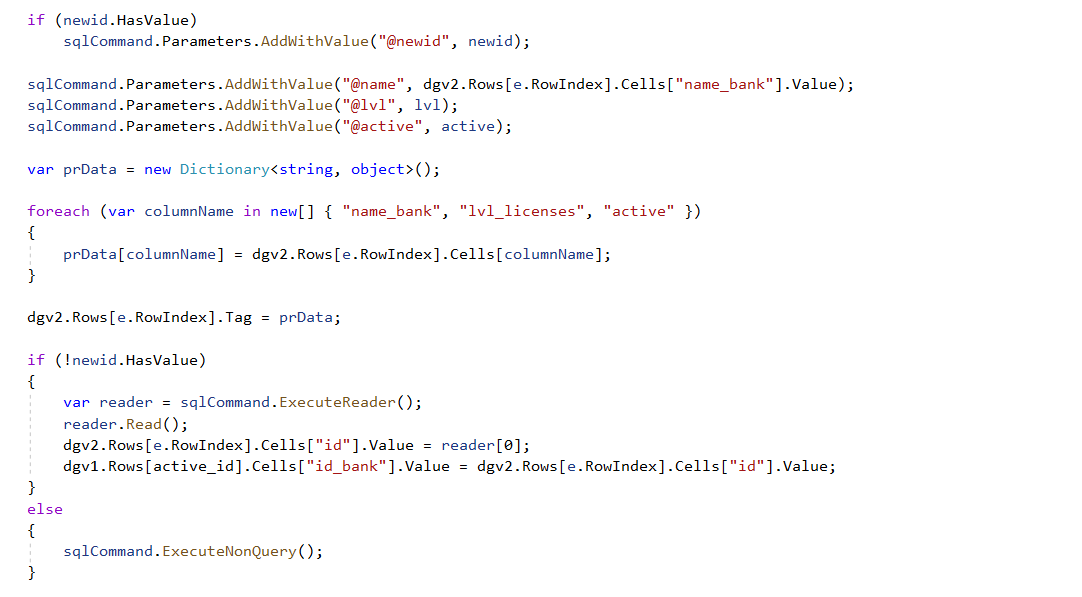




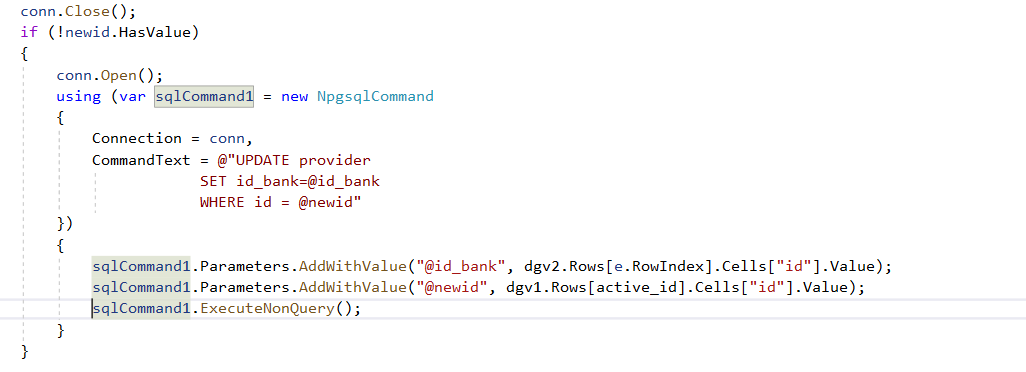


Проверка внутри программы происходит вызовам функции, хранящиеся в БД.

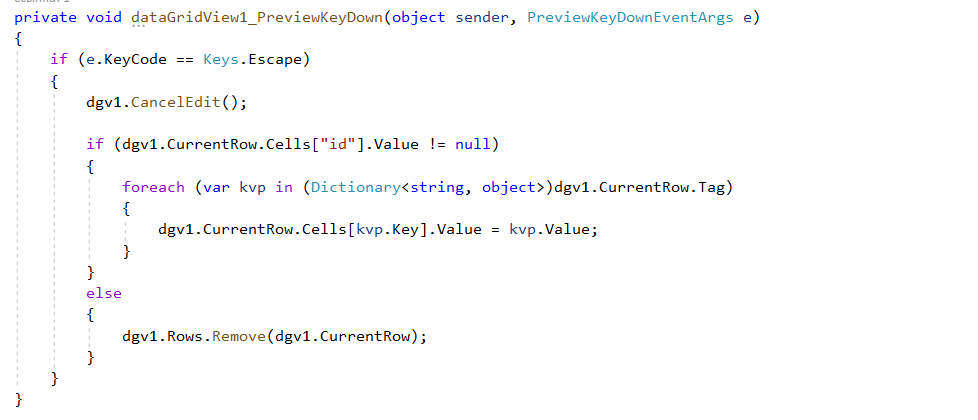
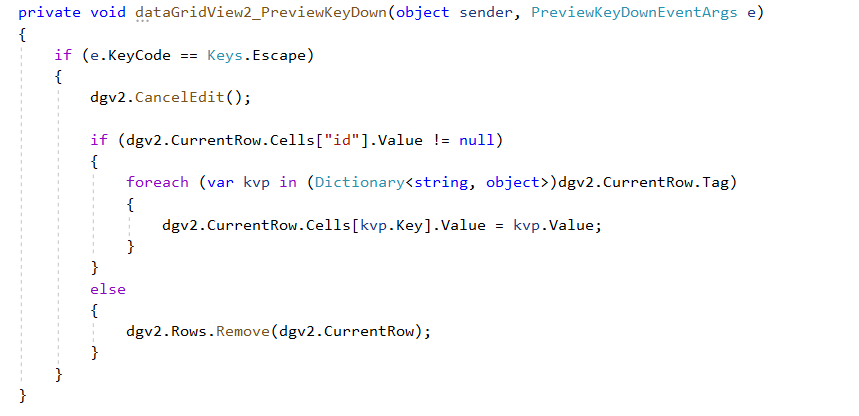
Остальные элементы кода аналогичны предыдущему методу



Так как мы можем добавлять и удалять записи связанные с первой таблицей нам необходимо как то отображать эту связь в БД, поэтому у нас есть поле в классе нормы отвечающее за номер активной выделенной строки.



Тесно связано с изменением и добавлением связано отмена изменений

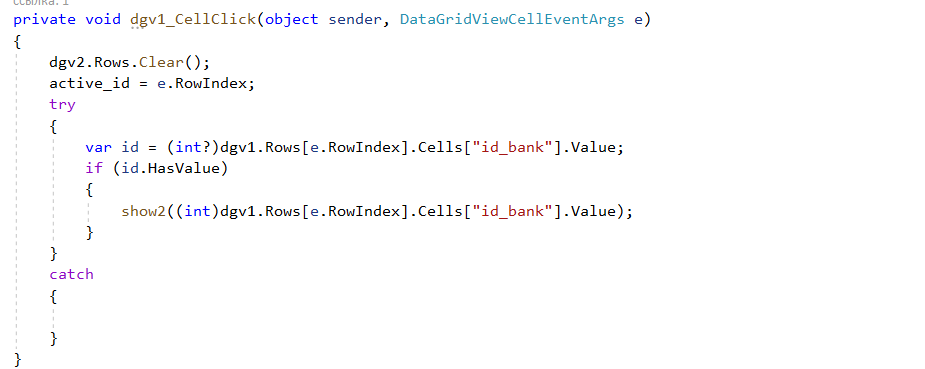
 

При нажатии клавиши ESC в поля элементов заполняются тегами их записей.

Удаление. Так как сам элемент удаляет из своих полей выбранную запись нам остается только отправить соответственный запрос в БД.

Так же нам необходимо отображать во втором представлении лишь те записи, которые связаны с первой таблицей



# 5 Тестирование программы

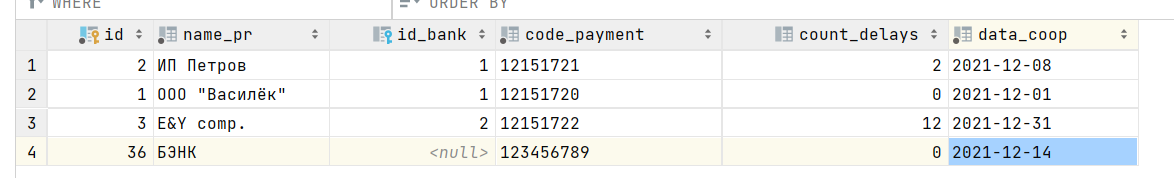
Тестирование программы можно разделить на 10 случаев:

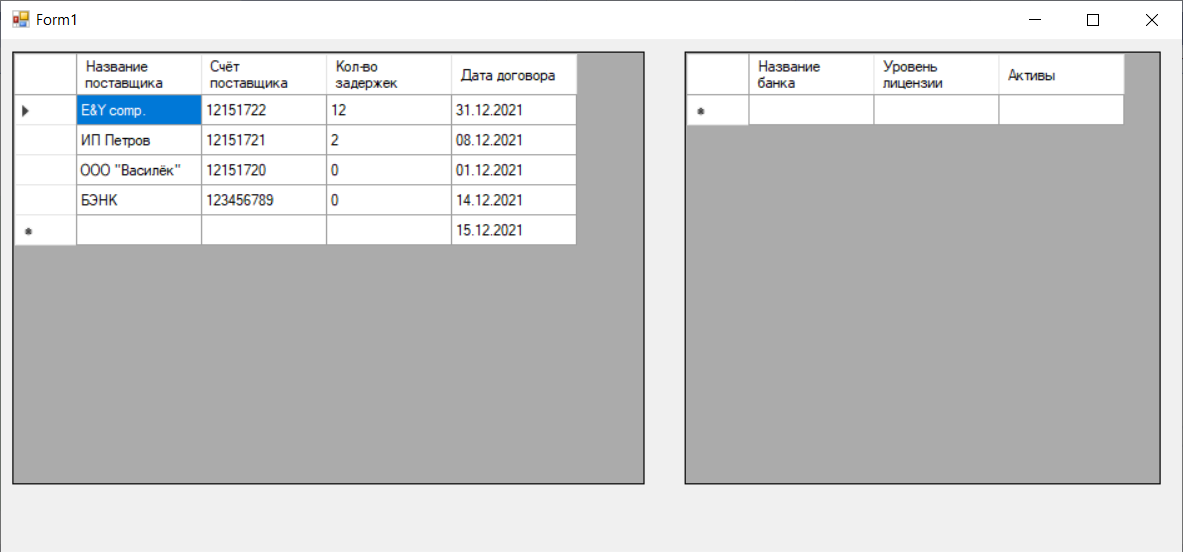
1. Изменение в таблице поля в таблице поставщики
2. Добавление поля в таблицу «Поставщики»
3. Удаления поля в таблице «Поставщики»
4. Попытка добавить поле в таблицу «Поставщики» с неправильными данными
5. Попытка изменить поле в таблице «Поставщики» с неправильными полями
6. Удаление поля в таблице «Банки»
7. Добавление поля в таблице «Банки»
8. Изменение поля в таблице «Банки»
9. Попытка добавить поле в таблицу «Банки» с неправильными данными
10. Попытка изменить поле в таблице «Банки» с неправильными полями

Тест 1(тип 1)

Входные данные:

{Поставщик1; 123456789; 0; 2021-12-14}

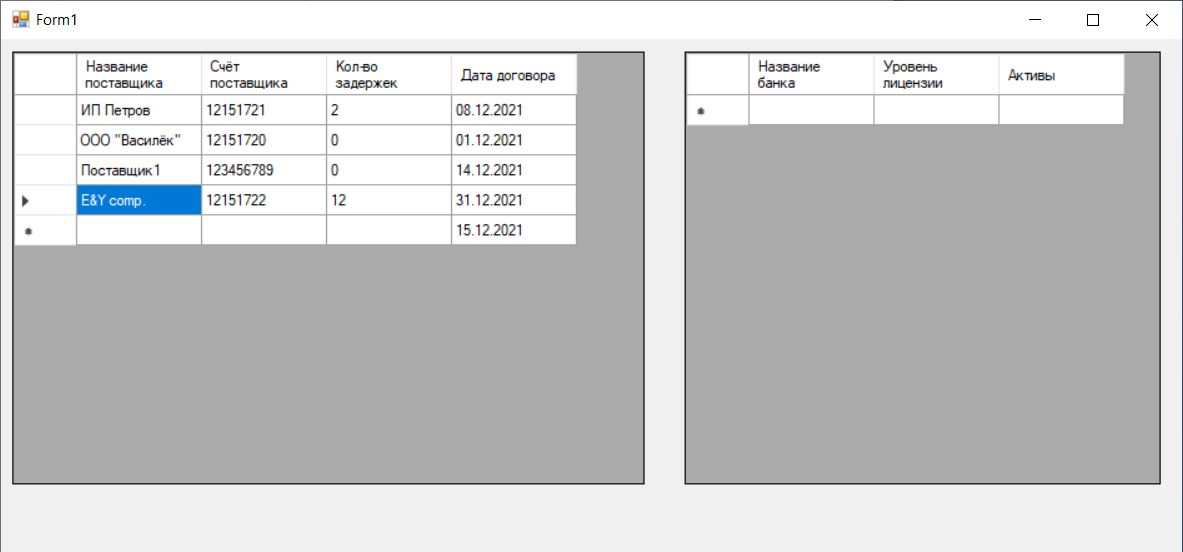


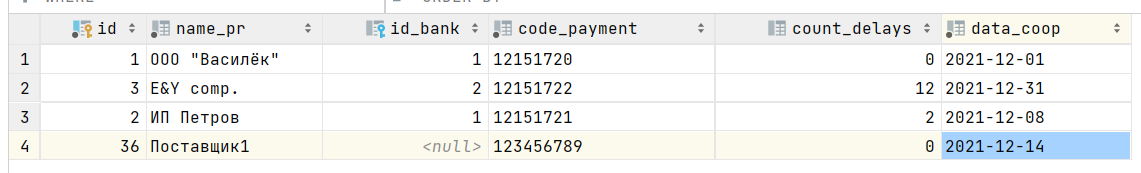


Ожидаемый результат:

Изменение полей заданной записи в БД

Выходные данные:

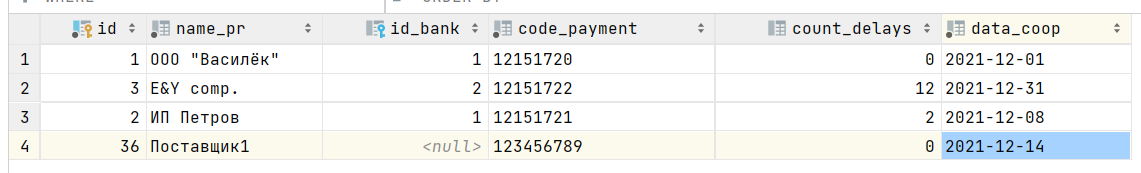


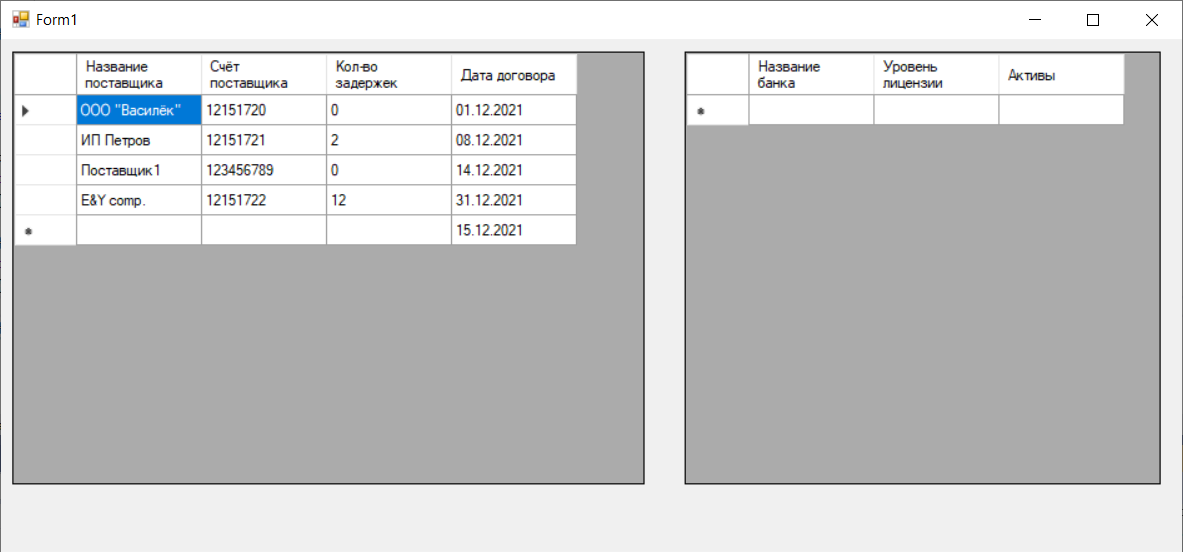


Тест 2(тип 2)

Входные данные:

{Поставщик2; 123456789; 2; 2021-21-15}

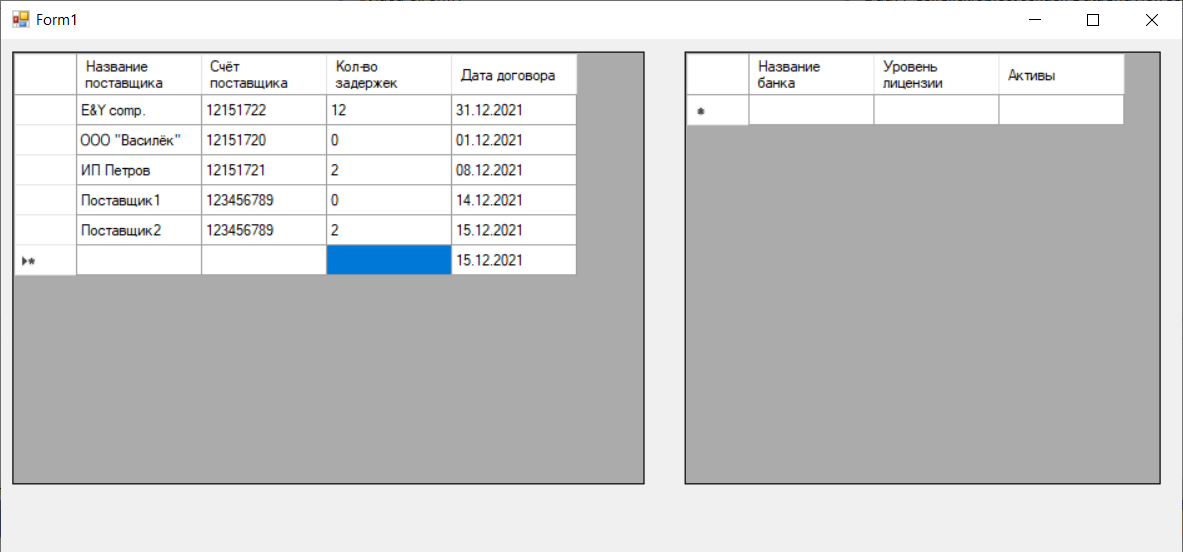


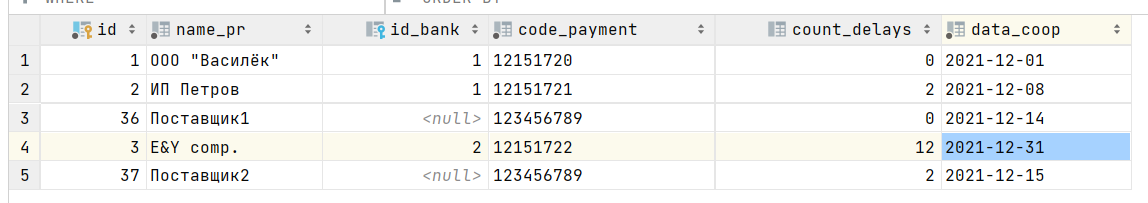


Ожидаемый результат:

Добавление нового поля в таблицу «Поставщики» в БД

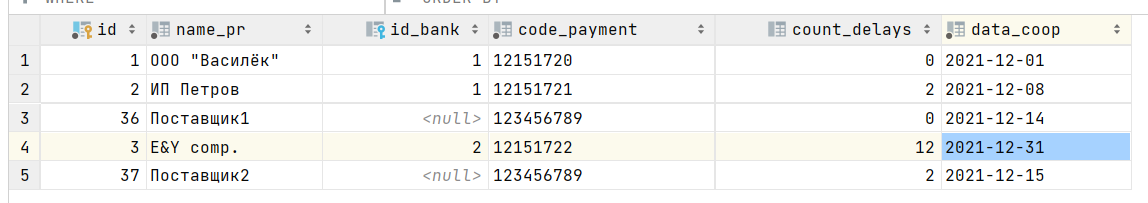
Выходные данные:

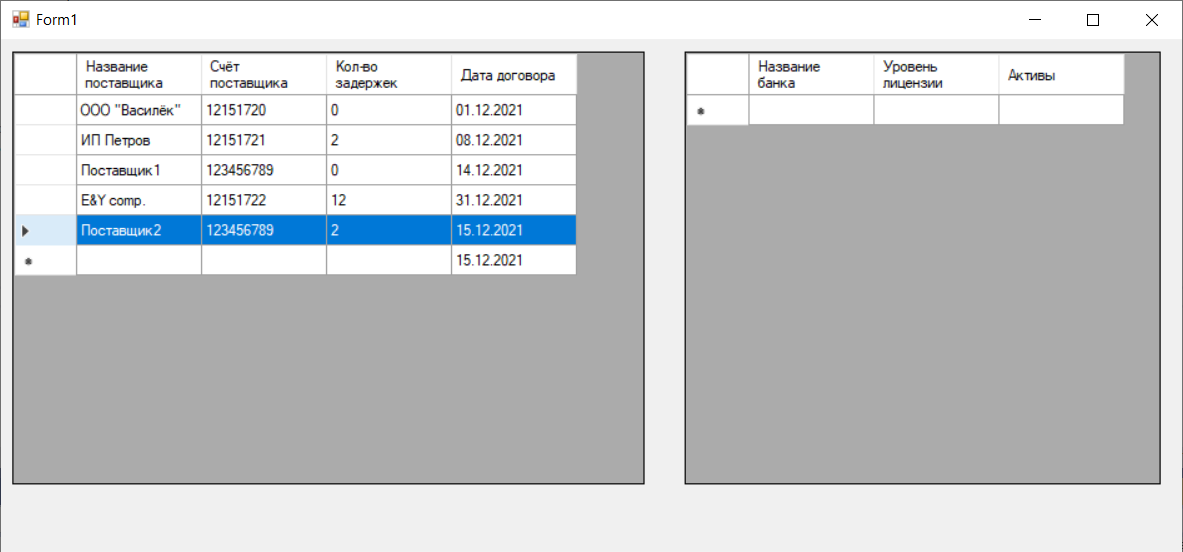




Тест 3(тип 3)

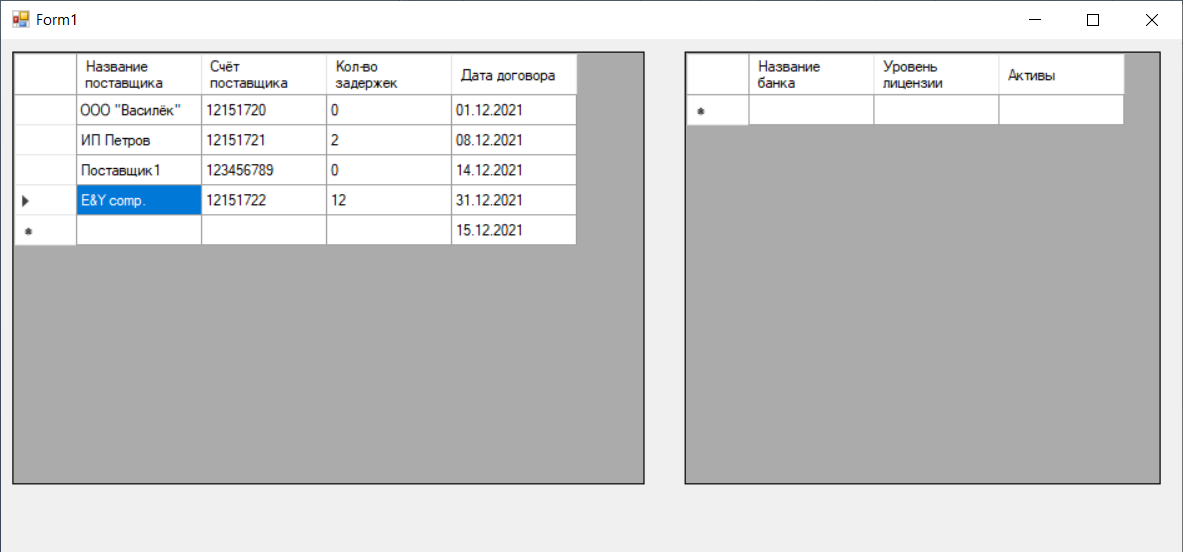
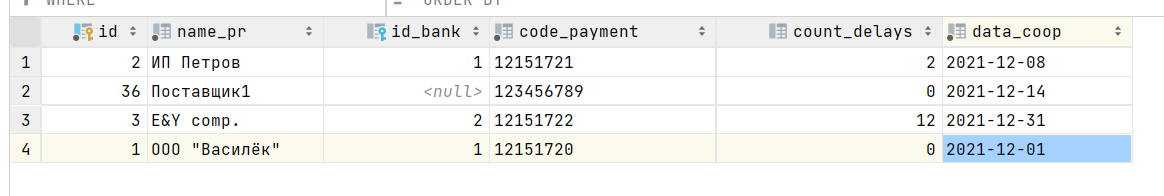
Входные данные:





Ожидаемый результат: удаление записи из таблицы «Поставщики в БД»

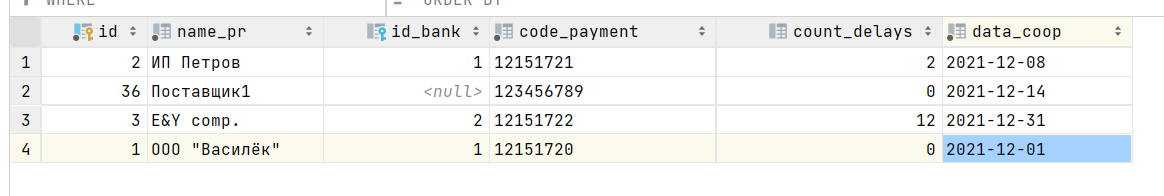
Выходной результат:

Тест 4(тип 4)

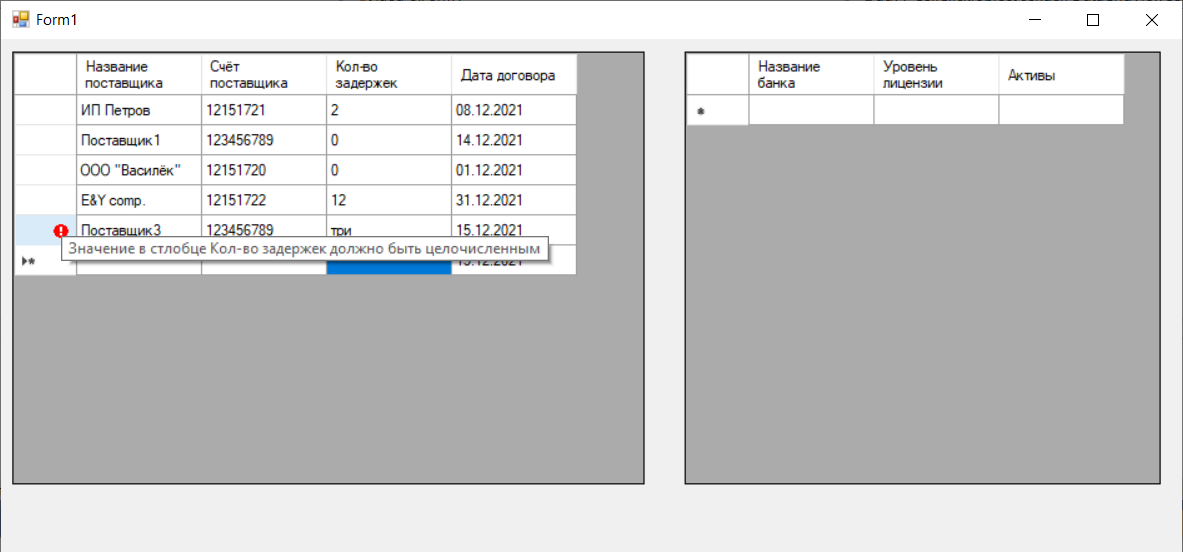
Входные данные:

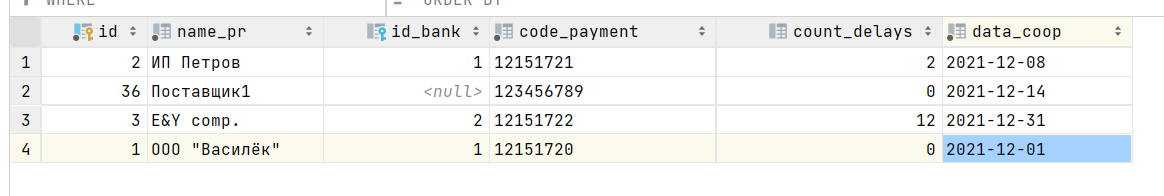
{Поставщик3; 123456789; три; 2021-12-15}



Ожидаемый результат: сообщение об ошибке

Выходные данные:

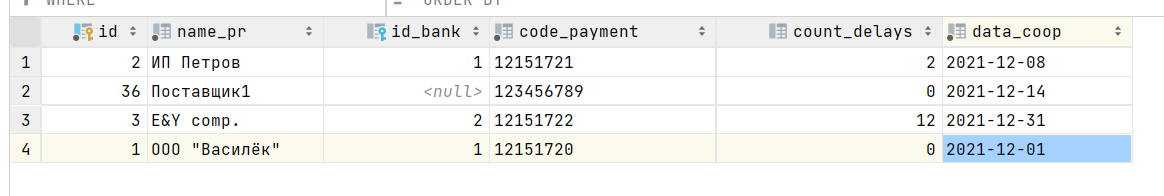


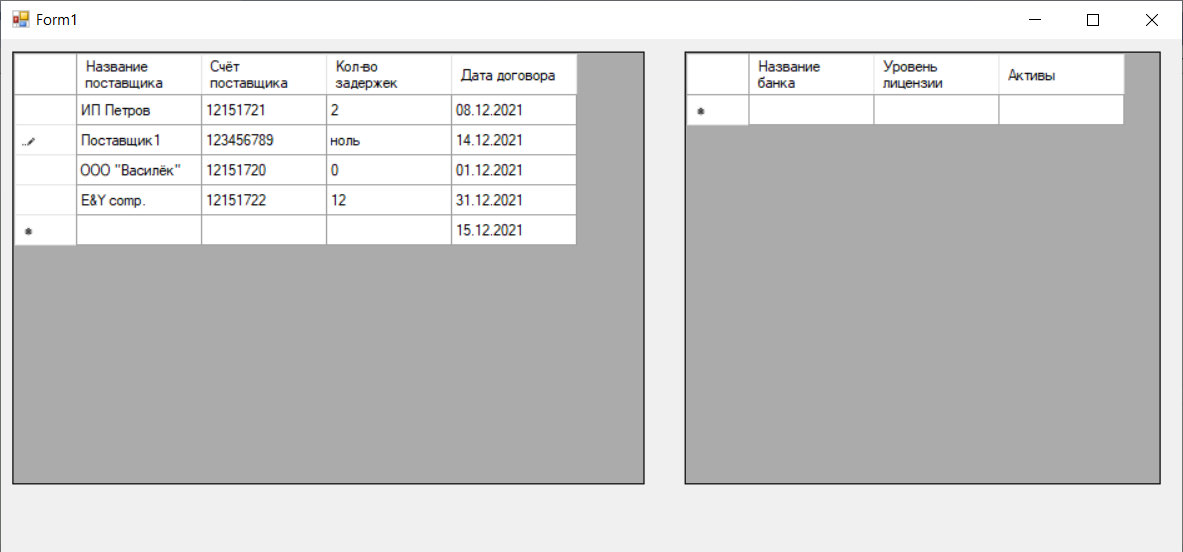


Тест 5 (тип 5)

Входные данные:

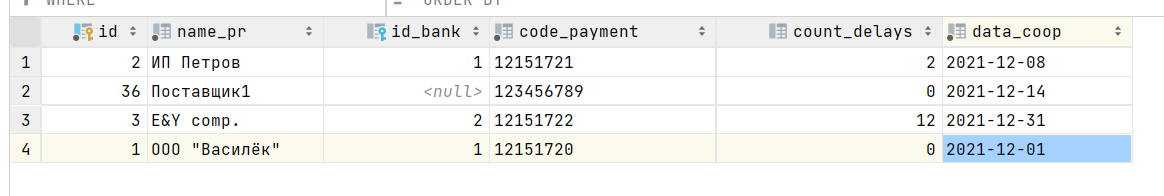
{Поставщик1; 123456789; ноль; 2021-12-14 }

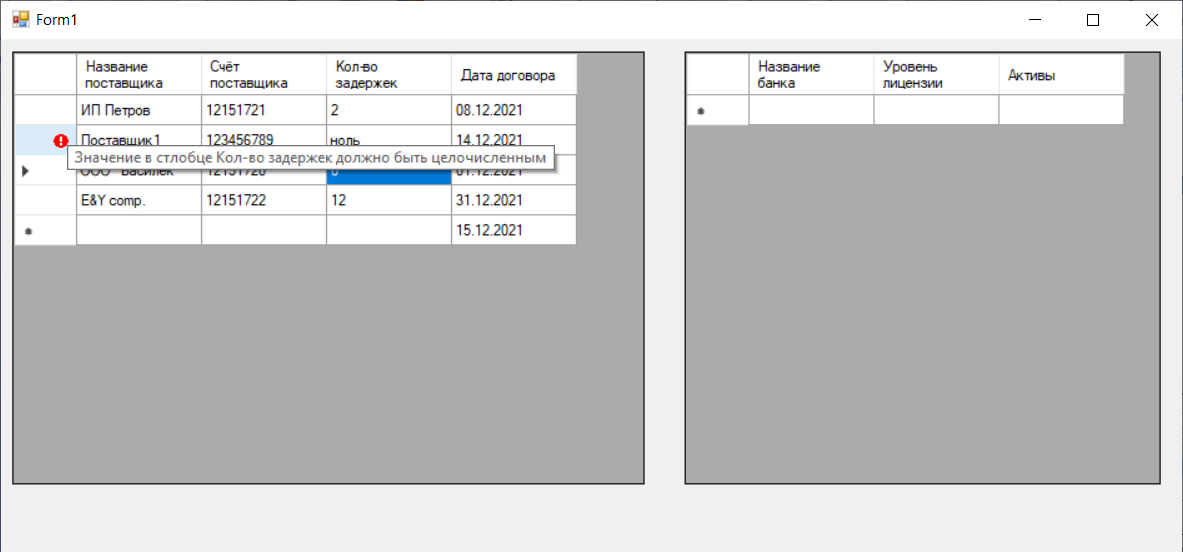




Ожидаемый результат: Сообщение об ошибке

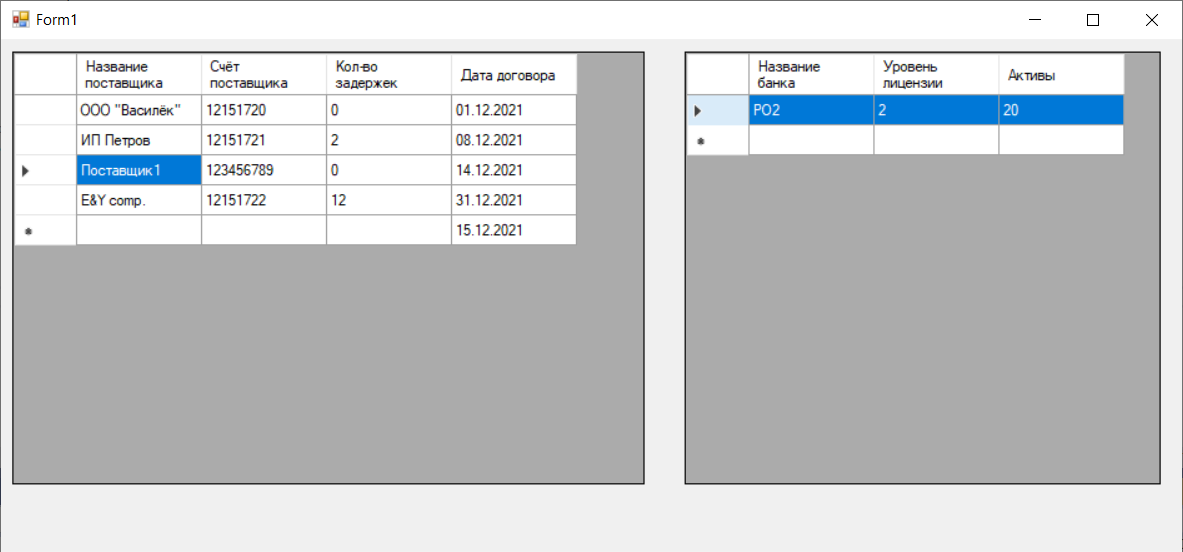
Выходные данные:





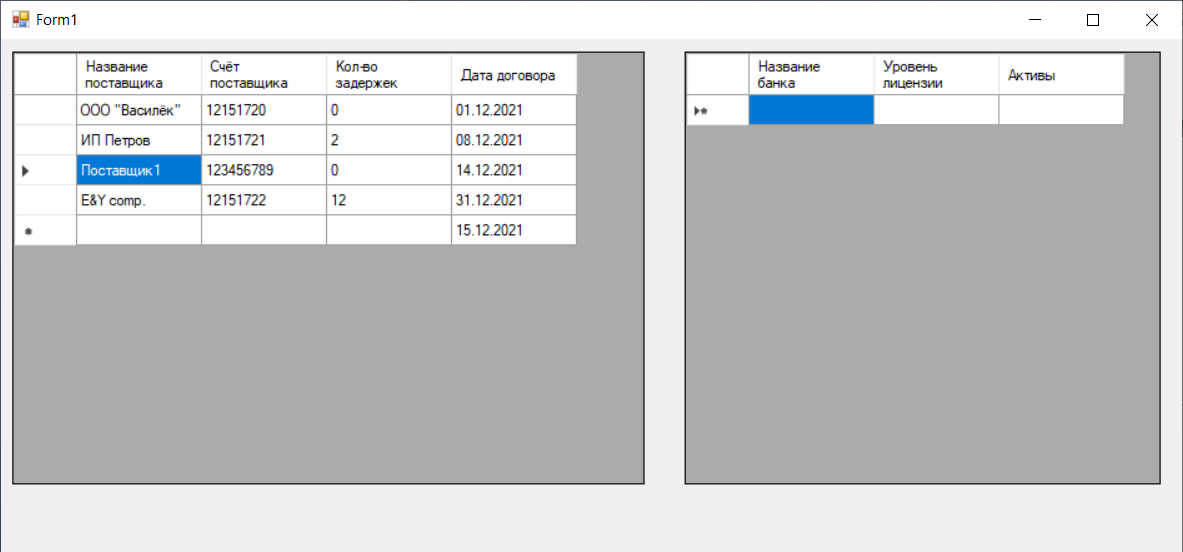
Тест 6 (тип 6)

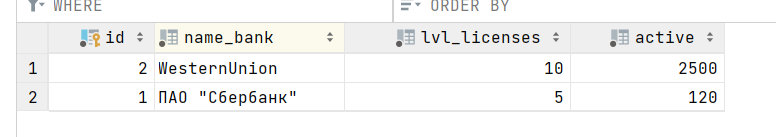
Входные данные:



Ожидаемый результат: удаление записи в таблице «Банки» в БД

Выходные данные:

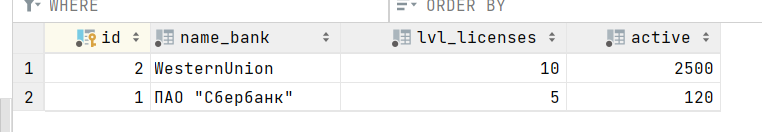
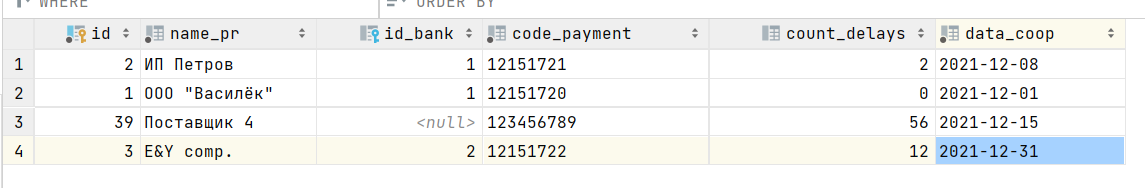


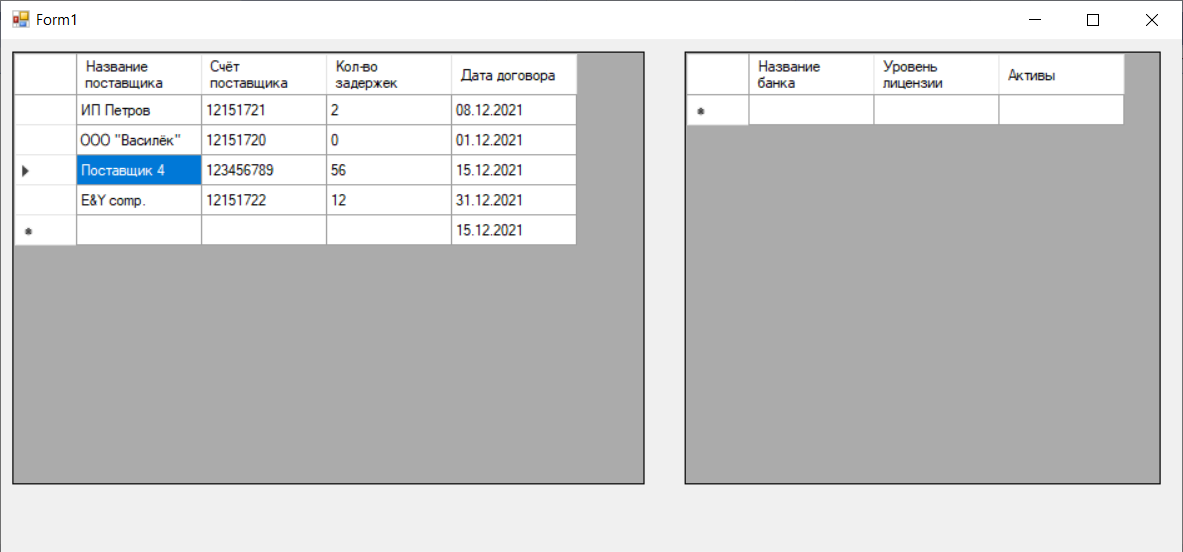


Тест 7(тип 7)

Входные данные:

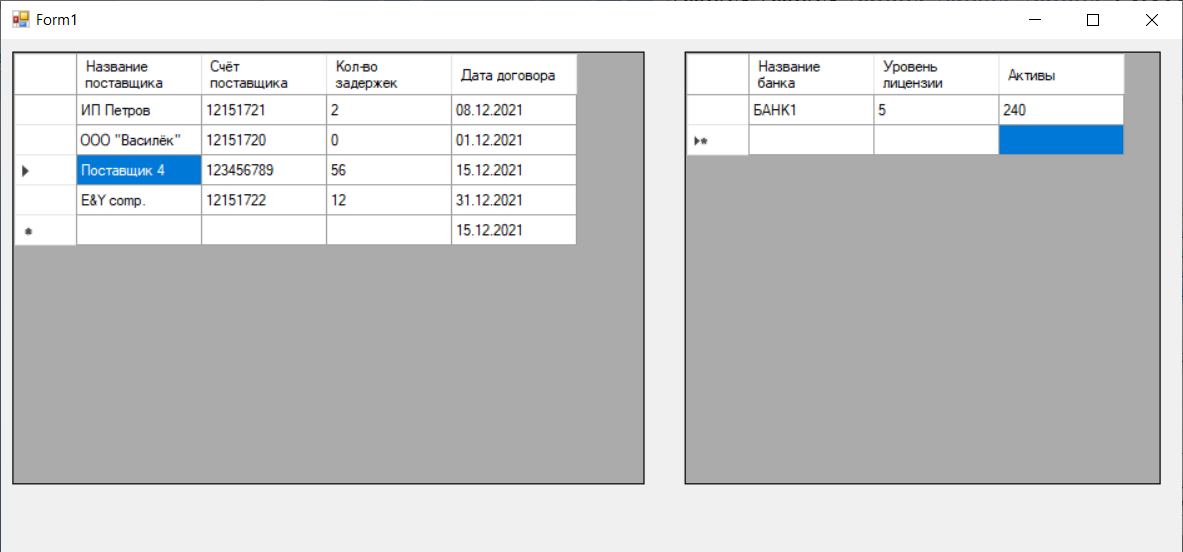
{БАНК 1; 5;250}

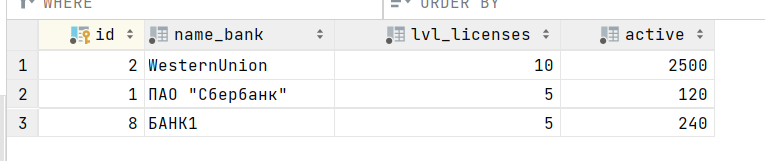


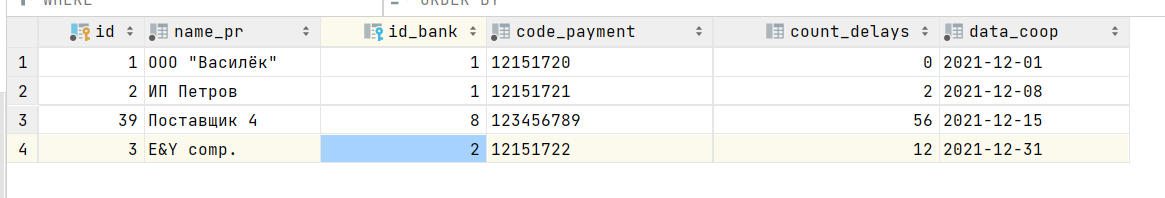


Ожидаемый результат: Появление новой записи в таблице «Банки» БД

Выходные данные:



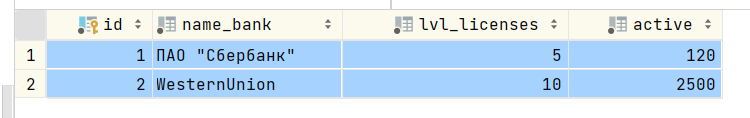


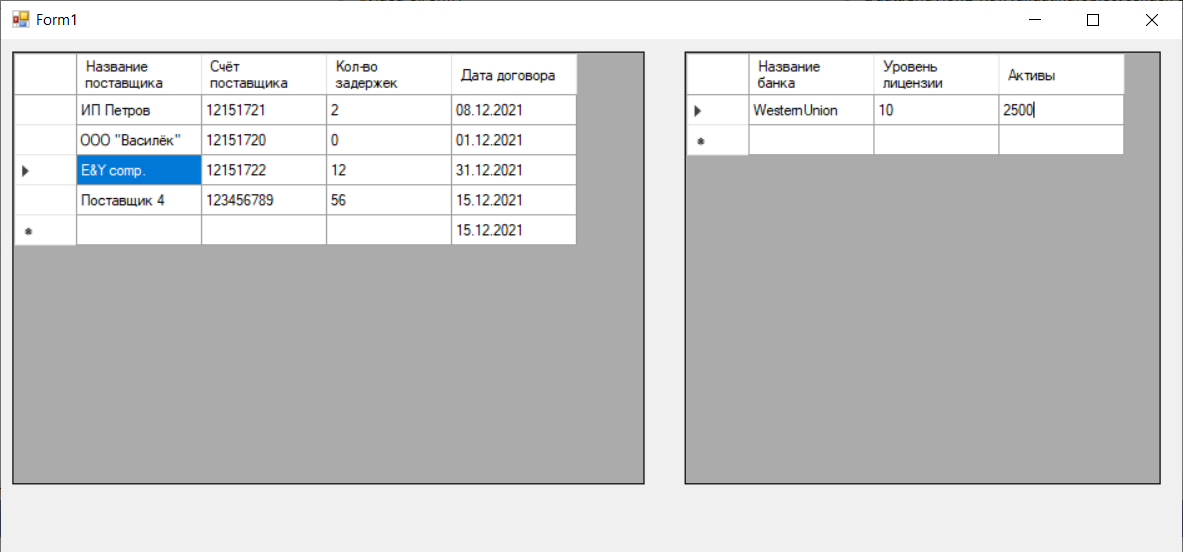


Тест 8(тип 8)

Входные данные

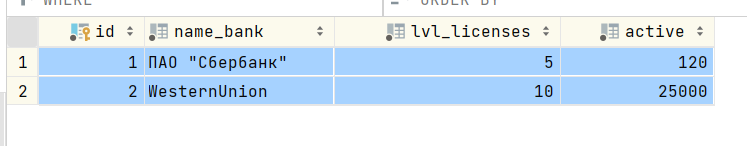
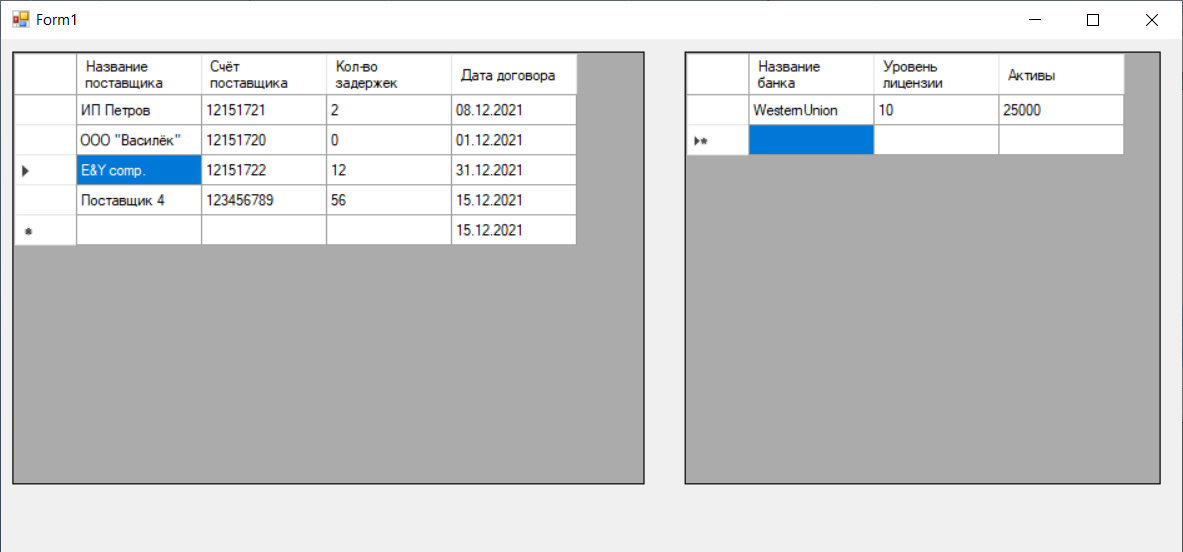
{WesternUnion; 10; 25000}





Ожидаемый результат: изменение строки в таблице «Банки» БД

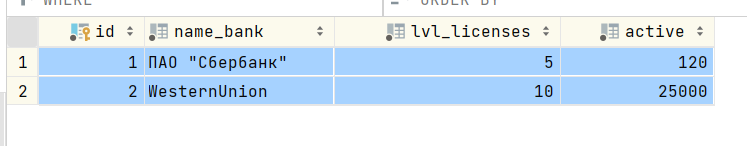
Выходные данные:

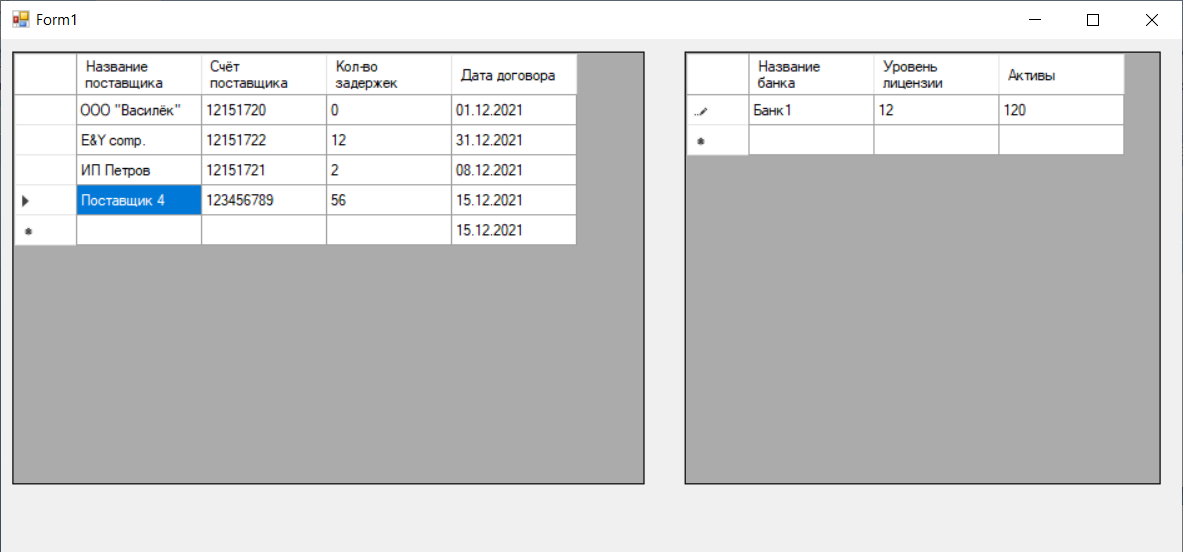


Тест 9(тип 9)

Входные данные:

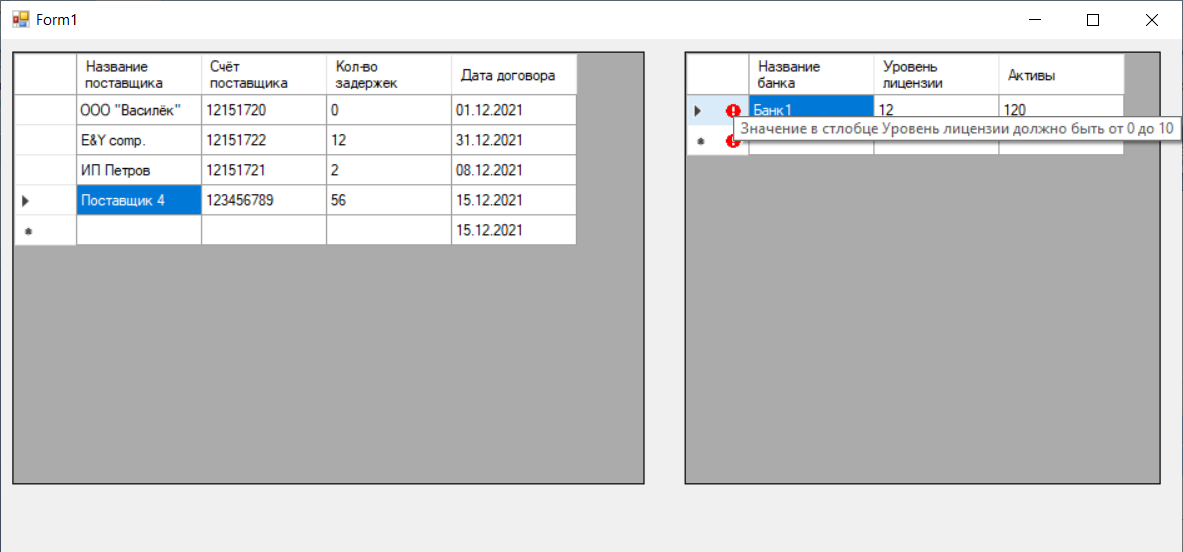
{Банк1; 12; 120}

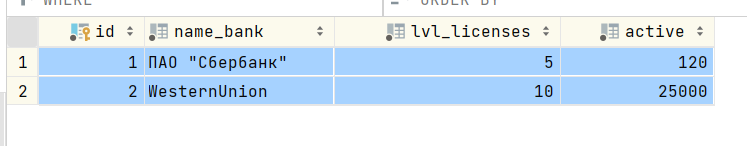




Ожидаемый результат: сообщение об ошибке

Выходные данные:

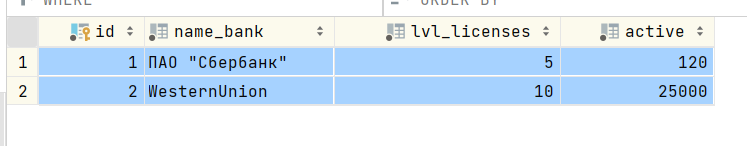


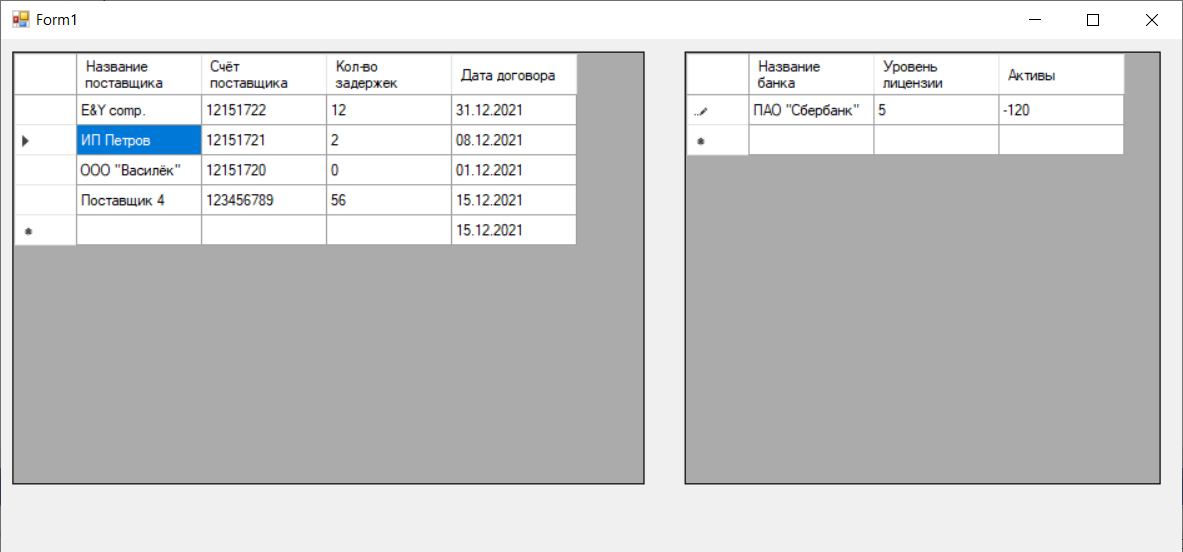


Тест 10(тип 10)

Входные данные:

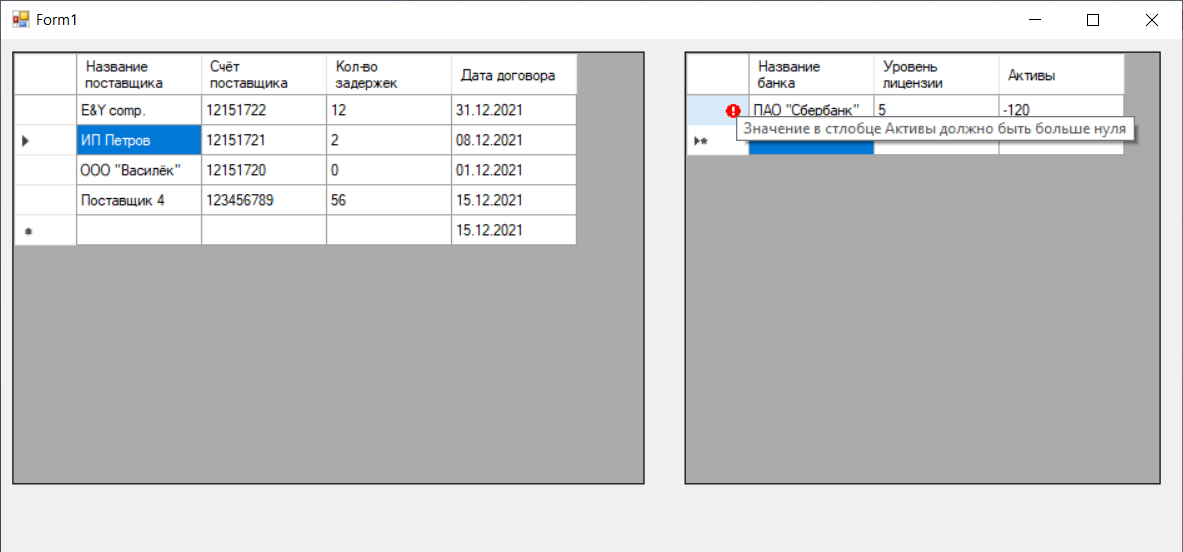
{ПАО «Сбербанк»; 5; -120}

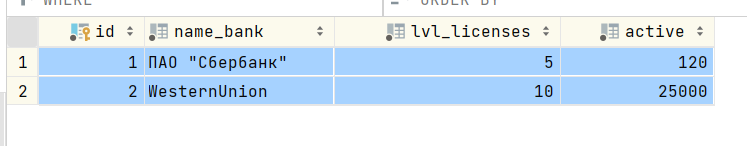




Ожидаемый результат: сообщение об ошибке

Выходные данные:





Вывод: Программа прошла все тесты и работает корректно.