**Плещев В.В.**

**Корректировка баз данных с различными *СУБД,***

**анализ, прогнозирование и аппроксимация данных в табличной и графической формах средствами Фреймворка *DBGridChart,***

***OLAP*-приложений *Экспресс\_ обработка.exe и***

***Экспресс\_ обработка\_таблиц\_Excel.exe.***

**Аннотация**

Предлагаемые средства (разрабочик Плещев В.В., E-mail: [Plehev@yandex.ru](mailto:Plehev@yandex.ru), гиперссылка для скачивания архивного файла *DBGridChart.zip* с *DLL*-библиотеками, текстами программам, документацией и демонстрационными примерами: <https://disk.yandex.ru/d/sI1VhBiSPzDp-A>) предназначены для автоматизации разработки программ на *C#* и на других языках программирования без оригинального программирования по формированию табличных и ленточных форм для ведения баз данных с любой *СУБД* с использованием источника данных *ODBC* (*SQL Server*, *MySQL, Access* и *Линтер* могут дополнительно подключаться по технологии *OLE DB*) в режиме непостоянного соединения с базой данных и немедленного (без промежуточного накопления изменений) обновления, сформированными Фреймворком из команды *SELECT*,командами *UPDATE, INSERT, DELETE* с временным подключением к базе данных только на период выполнения этих команд, а также для формирования без программирования заранее нерегламентированных аналитических таблиц, таблиц с прогнозами (сформированных различными методами), аппроксимацией, графиков и диаграмм различных типов для *OLAP*–анализа данных дата-аналитиками, по унифицированной технологии (с использованием управляющих элементов *C#* *dataGridView, Chart и SQL*) с много функциональным, унифицированным, адаптивным и комфортным для пользователей готовым интерфейсом.

Что-бы показать высокий уровень автоматизации программирования средствами Фреймворка для программиста, знающего *C#*, приведем пример текста процедуры обработки события для кнопки buttonavtomatform варианта автоматического формирования формы с одной таблицей для ведения базы данных с готовым интерфейсом:

private void buttonfoto\_Click(object sender, EventArgs e)

{

ClassBD\_dataGridView wbd = new ClassBD\_dataGridView(); // создание объекта (экземпляра) класса ClassBD\_dataGridView для таблицы "Изделия"

wbd.ConnString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=\"C:\\Демонстрационный пример\\Выпуск изделий.mdb\""; //строка подключения к БД;

wbd.Select = "SELECT Изделия.[Код изделия]@#, Изделия.[Наименование изделия]\*, Изделия.[Фото изделия в БД]#, Изделия.[Фото изделия в файлн], Изделия.[Код типа изделия]\*, [Типы изделий].[Наименование типа изделия] FROM [Типы изделий] INNER JOIN Изделия~ ON [Типы изделий].[Код типа изделия] = Изделия.[Код типа изделия]"; // текст запроса SQL со специальным символами-признаками после имен полей, если нужно корректировать БД: поле ключевое (@), обязательное (\*), некорректируемое (#) и после имени корректируемой таблицы в БД указывается символ ~

wbd.Podstavit\_v\_kolonki = "кти -#; SELECT [Наименование типа изделия], [Код типа изделия] FROM [Типы изделий] ORDER BY [Наименование типа изделия]"; // Подстановка кода типа изделия: кти – аббревиатура имени поля с кодом типа изделия для подстановки, список кодов исчерпывающий (-) или нет (+) и выводится в ячейке таблицы (#) или в первой строке экрана дисплея.

wbd.Zagruzit("29456"); //Загрузить данные в таблицу dataGridView129456 29456 - случайные символы для уникальности имени таблицы т.к. параметры настройки, макета при их сохранении, привязываются к имени таблицы.

}

Из примера видно, что только две команды *SQL* (wbd.Select, wbd.Podstavit\_v\_kolonki) из пяти можно отнести к оригинальным. На этом заканчивается программирование на языке *C#*!

В приведенном примере, информации достаточно для написания программистом аналогичных программ без изучения документации.

В результате выполнения этой процедуры на экран выведется таблица с готовым интерфейсом для пользователя (рис. 1):



**Рисунок 1 – Просмотр и корректировка таблицы с изображениями.**

Пользователь при выполнении приложения может выбрать нужные для работы элементы интерфейса и сохранить этот выбор для следующих сеансов работы, а также может воспользоваться средствами конструктора макетов для изменения интерфейса (например, цвет экрана, размер шрифта, заголовки колонок, порядок вывода колонок на экран).

При использовании редкого варианта программного формирования формы с несколькими таблицами к этим командам добавляются процедуры обработки нескольких событий для каждой таблицы в которых нужно написать по одной команде вызова соответствующего метода Фреймворка. Описание их с примерами можно найти в документации.

При разработке программ на других языках программирования отличных от *C#*, используется готовое приложение–посредник *Экспресс\_обработка\_таблицы.exe* которому передается строка аргументов со значениями свойств классов Фреймворкадля формирования обращения к средствам Фреймворка по созданию формы с одной таблицей с готовым интерфейсом.

Примеры таких программ для разных языков программирования приведены в конце аннотации.

Использование *OLAP*-приложения *Экспресс\_обработка.exe* вообще исключает программирование полностью: строки подключения к БД, сформированные конструктором запросов Фреймворка, команды *SELECT*, другие команды *SQL*, включая выполнение хранимых процедур *SQL*, процедуры (группа из нескольких команд *SELECT* и *SQL*)вынесены в отдельный текстовый файл. Работа с этим файлом ведется в табличной форме с готовым интерфейсом, включая формирование и выполнение готовых запросов, команд *SQL* и процедур самим пользователем без использования языка программирования.

Во Фреймворке *DBGridChart* допускается работа в режиме чтения с электронными таблицами *Excel* в приложениях *Экспресс\_обработка.exe* и *Экспресс\_обработка\_таблиц\_Excel.exe*.

В таблице запросов для приложения *Экспресс\_обработка.exe* указывается строка подключения, аналогичная для *Access* c добавлением параметров с описанием особенностей *Excel* различных версий. Текст этой строки формируется автоматически при открытии пользователем Excel-файла.

Например:

Provider=Microsoft.Jet.OLEDB.4.0; Data Source="C:\Экспресс обработка\Подразделения.xls"; Extended Properties='Excel 8.0;HDR=Yes;IMEX=1'

и текст запроса, например: *SELECT \* FROM [Лист1$]*

Имеется автономный вариант работы пользователя с таблицами *Excel* с использованием приложения *Экспресс\_обработка\_таблиц\_Excel.exe*. Пользователь выбирает из окна поиска файлов нужный *Excel*-файл, выбирает нужный лист из, автоматически сформированного, списка листов в этом файле, выбранный лист загружается в таблицу *DataGridView*, которая отображается на экране с готовым интерфейсом Фреймворка *DBGridChart*.

Далее, пользователь формирует различные аналитические таблицы, прогнозы и отображает результаты в графической форме. Это позволяет объединить возможности *Excel* и Фреймворка *DBGridChart*.

Для конечного пользователя нет необходимости изучать документацию из-за наличия всплывающих подсказок, справочной информации расположенной на формах или вызываемой по кнопкам или по гиперссылкам на страницы нужных сайтов.

Далее, приводится описание основных возможностей и особенностей Фреймворка.

Результаты выполнения исходной команды *SELECT*помещаются адаптером *C#* *DataAdapter* в таблицу  *DataTable* и эта таблица указывается в качестве источника данных *DataSource* для таблицы *dataGridView* с которой, далее работает Фреймворк. Это максимально быстрый для *C#* способ загрузки данных по запросу *SELECT* из базы данных в таблицу *dataGridView* (примерно, от 200000 и выше записей в секунду в зависимости от типов компьютера, *СУБД* и числа колонок в таблице).

Сортировка по нескольким колонкам релизована методом *C# dataView.Sort* для таблицы (более 1000000 строк в секунду) ифразой *ORDER BY* команды *SELECT* для базы данных. Выбор режима сортировки (в таблице или в БД) указывается в окне конструктора макетов. При сортировке записей в базе данных пользователю не нужно знать язык *SQL* – он только, как и при сортировке таблицы, указывает мышкой колонки и порядок сортировки, а формирование фразы *ORDER BY* выполняется автоматически.

Фильтрация данных является одной из базовых процедур обработки данных и во Фреймворке ей уделено особое внимание.

Фильтрация релизована методами *C# dataView.RowFilter* для таблицы ифразой *WHERE* команды *SELECT* для базы данных. Выбор режима фильтрации (для таблицы или БД) указывается в окне конструктора фильтров.

При фильтрации записей в базе данных пользователю не нужно знать язык *SQL* – он только задает условие фильтрации, а перевод условия на язык *SQL* выполняется автоматически.

Конструктором фильтров Фреймворка можно сформировать стандартный, оперативный и оригинальный фильтры.

Стандартный фильтр формируется для одной выбранной колонки диалоговыми средствами. Сформированное условие фильтра, по желанию пользователя, можно указать для фильтрации самой таблицы *dataView.RowFilter* или данных непосредственно в базе данных фразой *WHERE*, котороя добавляется в текст исходной команды *SELECT* и повторной автоматической загрузки отфильтрованных данных в таблицу. Фильтрация данных в базе данных позволяет уменьшить объем и время загрузки данных в таблицу.

Оперативный фильтр расположен непосредственно в строке интерфейса и состоит из надписи с наименованием операции сравнения, выбранной из стандартного фильтра, и поля с шаблоном поиска. При каждом изменении (ввода или удаления символа) пользователем этого поля сразу производится фильтрация строк *dataView.RowFilter* по указанной команде сравнения и текущему значению шаблона без обращения к стандартному фильтру.

Например, можно оперативно фильтровать по первым буквам в любой колонке таблицы (значения нетекстовых колонок автоматически переводятся в текстовый формат перед сравнением).

Оригинальный фильтр формируется произвольным образом с использованием имен колонок таблицы или полей базы данных и значений полей, различных операций и функций, допускаемых в условии фильтра *RowFilter* или фразы *WHERE*  указанных с их описанием в конструкторе. Фактически, пользователь может сформировать любой сложности фильтр с использованием нескольких колонок таблиц или полей базы данных при фильтрации записей в базе данных.

Сформированные стандартный и оригинальный фильтры можно применять отдельно или вместе, объединенных логическими операциями *AND*, *OR*.

Тексты с условиями в фильтрах можно сохранить в текстовом файле для повторного использования.

Фильтрация производится быстро – более 1000000 строк в секунду.

Таким образом, во Фреймворке непосредственно используются стандартные высоко скоростные средства *C#* и *SQL* загрузки данных из базы данных в таблицы (*DataAdapter, SELECT*), сортировки (*dataView.Sort, ORDER BY*) и фильтрации строк в таблице или записей в базе данных (*dataView.RowFilter, WHERE*), что позволяет использовать Фреймворк при работе с реальными большими базами данных.

Для формирования запросов и команд *SQL*разработан специальный визуальный конструктор со справочной системой и с примерами. Информация о таблицах, запросах *SELECT* и их полях берется автоматически конструктором из базы данных, указанной в строке подключения к базе данных, и предоставляется пользователю для формирования команд *SELECT*и *SQL*.

Наличие конструкторов команд *SQL* и фильтров позволяют пользователю в диалоговом режиме мышкой формировать нужные команды и фильтры.

Практически, программисту для каждой табличной формы нужно задать только текст одной команды *SELECT*для просмотра и корректировки данных в базе данных и команд *SELECT*для подстановки кодов наименований из таблиц-справочников с наименованиями кодов (справочники изделий, предприятий, подразделений, сотрудников, единиц измерений и др.), а все остальное, включая, настраиваемый интерфейс пользователя, реализуется автоматически средствами классов Фреймворка.

Диаграммы полностью формируются самим пользователем нужного типа и содержания.

Корректировка текущей записи выполняется командой *UPDATE*немедленно после каждого изменения значения ячейки в таблице *DataGridView* при условии задания значений для полей, обязательных для заполнения.

Добавление записи определяется автоматически (отсутствием записи с введенными значениями ключевых полей) и производится немедленно после окончания ввода записи командой *INSERT*.

Сразу после корректировки или добавления записи производится контрольное чтение командой *SELECT* и отображение на экране обновленных данных для подтверждения корректировки или добавления и проверки пользователем правильности внесенных изменений.

Удаление, выделенных мышкой, записей выполняется командой *DELETE*.

Сообщения об ошибках, обнаруженных фреймворком или *СУБД*немедленно выводятся на экран в окне сообщений. Ошибки могут быть исправлены пользователем немедленно.  
Создается иллюзия постоянного соединения с базой данных.

Нужно отметить, что при большом числе строк в запросе рекомендуется в запросе *SELECT* использоватьфразы *WHERE* и/или *GROUP BY* для фильтрации (включая диапазон порядковых номеров загружаемых записей) и/или группировки строк, что существенно уменьшает объем выводимой информации для пользователя и сокращается время работы.

***Основные возможности Фреймворка.***

Копирование, сортировка по нескольким колонкам, закрепление и скрытие колонок на экране, изменение ширины и  
местоположения колонок, развертывание таблицы, с переносом кнопок в начало экрана, на весь экран и обратное свертывание.  
Подстановка значений кодов из справочников с наименованиями кодов (аналог подстановки в *Access*).

Допускается работа с изображениями (содержанием графических файлов) в полях БД: загрузка в поля БД изображений из графических файлов, удаление изображений из полей, экспорт изображений из полей в отдельные графические файлы. Изображение выводится из текущей ячейки на весь экран правой кнопкой мышки. В этом варианте графические файлы можно удалить, но объем базы данных существенно увеличивается за счет хранения содержания файлов.

Во втором варианте (связанных файлов) в БД можно сохранять вместо содержания только полные наименования файлов любых типов (не только графических) и открывать их из текущей ячейки таблицы правой кнопкой мышки. Это позволяет сократить объем БД и открывать любые файлы, а не только графические, например, аудио-, видео-, *Doc-*, *PDF-*, *Excel-*, *EXE-*файлы. Такие файлы можно разместить в отдельные папки на сервере и не менять их расположение (диск и систему папок – полный путь к файлам) после загрузки в БД их полных наименований. Но если изменения необходимы, то для этого предусмотрены программные средства замены в пакетном режиме текущего местоположения нужных связанных файлов на новое местоположение.

При выводе таблицы с большим числом колонок пользователь может развернуть таблицу в новую ленточную таблицу.  
Одна полоса может содержать до пяти ячеек. Это позволяет отказаться от необходимости разработки программ формирования разнообразных простых не табличных форм с большим числом колонок.

Визуальное формирование непосредственно пользователем из исходной таблицы, с фильтрацией строк и колонок, различных расчетных, аналитических колонок, итоговых таблиц, итоговых ступенчатых таблиц-отчетов, итоговых перекрестных таблиц, таблиц с прогнозами (экстраполяцией), аппроксимацией и диаграмм разных типов.

Изображения графиков и диаграмм можно сохранить в отдельных графических файлах нужного типа (*Jpeg, Png, Bmp, Gif, Tiff*).

Визуальное формирование запросов, команд *SQL* с синтаксическим контролем, просмотр любых таблиц и запросов в БД, специально разработанным, конструктором.

Выполнение любых команд *SQL*, включая команду *EXEС* для выполнения хранимых процедур.

Экспорт, импорт и распечатка таблиц и изображений.

Состав кнопок и полей интерфейса, включая иконки на кнопках и тексты подсказок, задаются в классе *Hablon* и могут изменены программистом или самим пользователем при выполнении приложения (состав элементов интерфейса).

***OLAP*-приложение *Экспресс\_обработка.exe* позволяет:**

вынести описания запросов из исходных текстов программ в отдельный текстовый файл, что исключает необходимость в разработке оригинальных программ (с обработкой кнопок, событий и др.);  
запросы и команды *SQL*можно объединять в процедуры и при их запуске автоматически выполняются все запросы, команды *SQL* и внутренние процедуры, которые указаны в процедуре;

формировать новые или изменять существующие запросы *Select*, команды *SQL*и процедуры, формировать дополнительные расчетные колонки самим пользователем конструктором запросов и сохранять их в текстовом файле запросов для дальнейшего использования;

в сложных случаях, когда одним запросом *Select* нельзя реализовать потребности аналитика данных, можно использовать процедуру из нескольких взаимосвязанных запросов и команд *SQL* (для создания и удаления временных таблиц в базе данных, выполнения существующих запросов и процедур и др.);

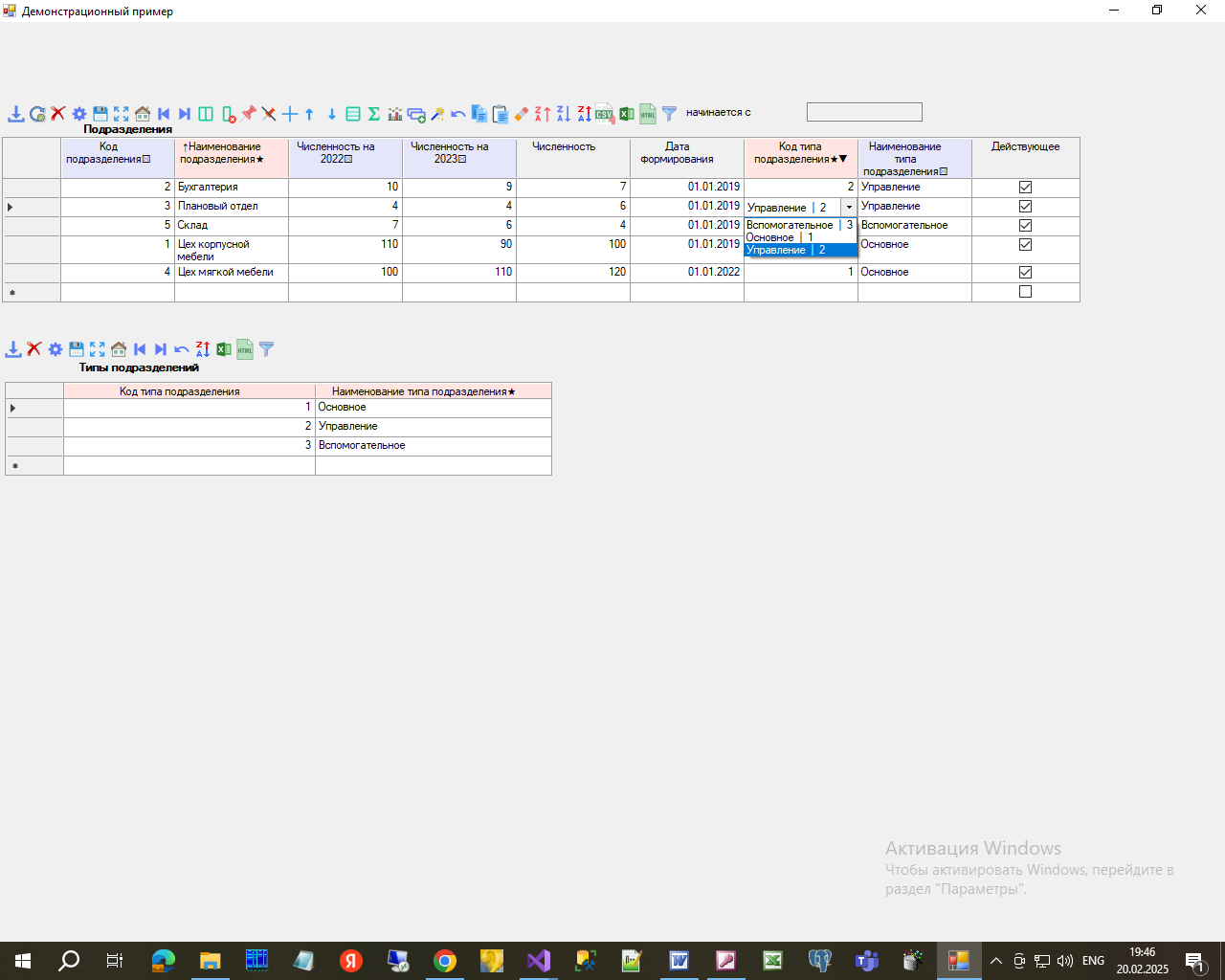
оперативно корректировать пользователям базы данных с различными *СУБД*и формировать аналитические таблицы, таблицы с прогнозом (экстраполяцией), аппроксимацией, диаграммы и графики без программирования;

предусмотрены средства администрирования и защиты от несанкционированного использования баз данных путем указания логинов, паролей и доступных для пользователей баз данных и их таблиц и запросов при регистрации пользователей и скрытия строк подключения к базам данных для всех пользователей (в таблице запросов указываются не строки подключения, а имена баз данных, указанные администратором баз данных).

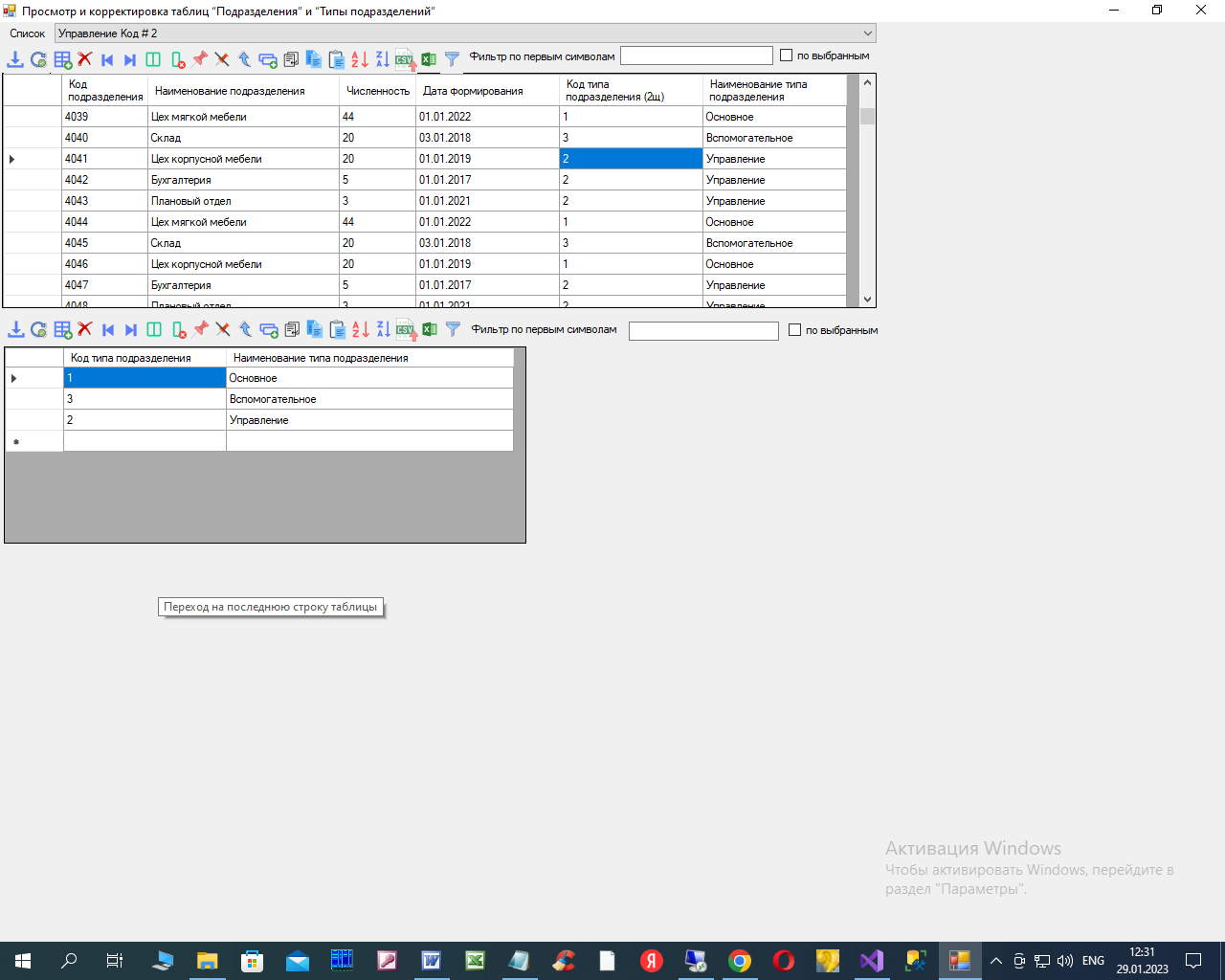
Фреймворк можно использовать при разработке программ на *C++*, *F#*, *Visual Basic, Python, Java* и на других языках программирования, используя готовое приложение *Экспресс\_обработка\_таблицы.exe*, выполняемое из командой строки,которому передается строка аргументов-параметров со значениями свойств класса *ClassBD\_dataGridView* .

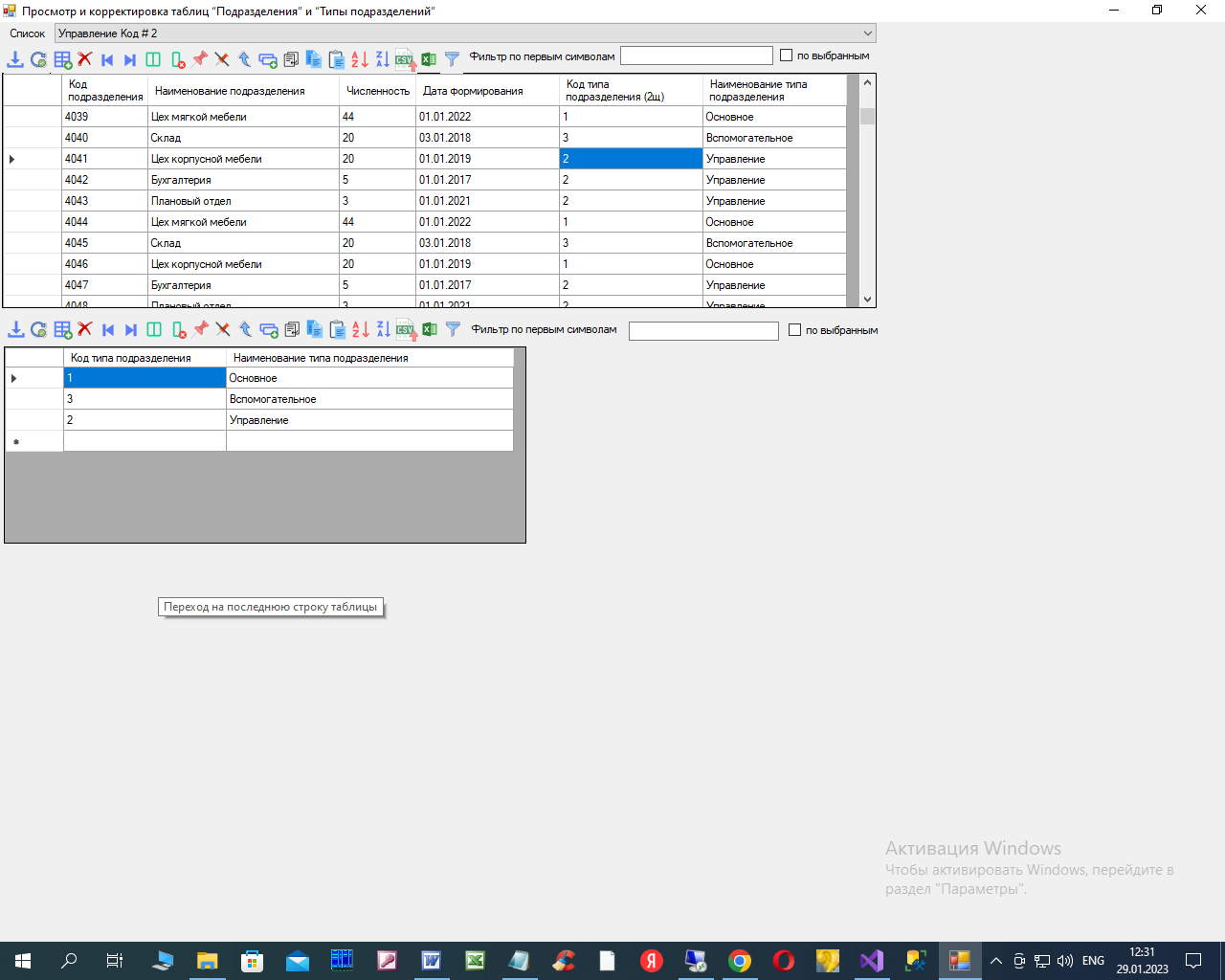
Предлагаемые средства прошли тестирование при работе с базами данных с различными *СУБД*и используются в учебной и практической деятельности.

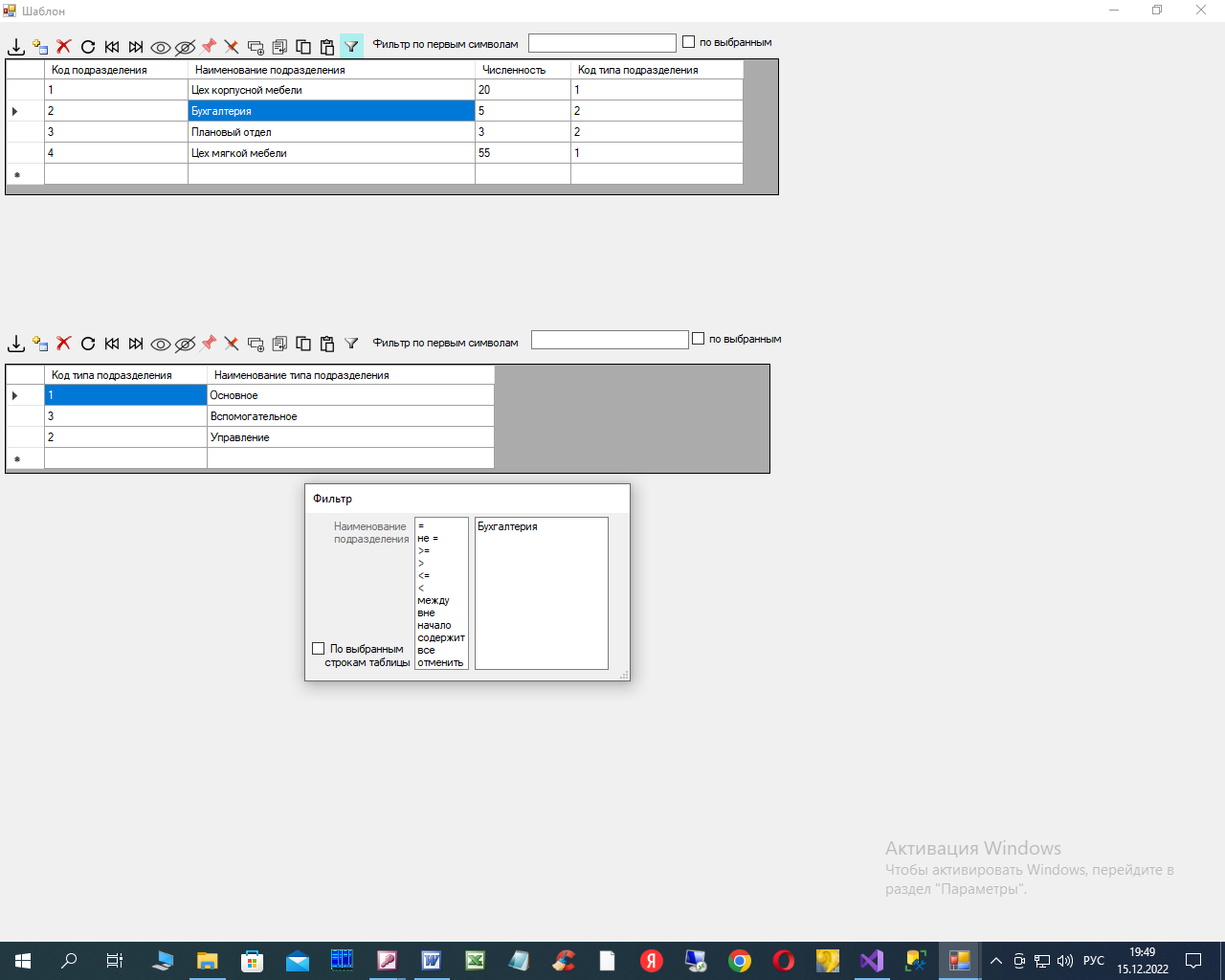
***Примеры элементов многофункционального интерфейса.***

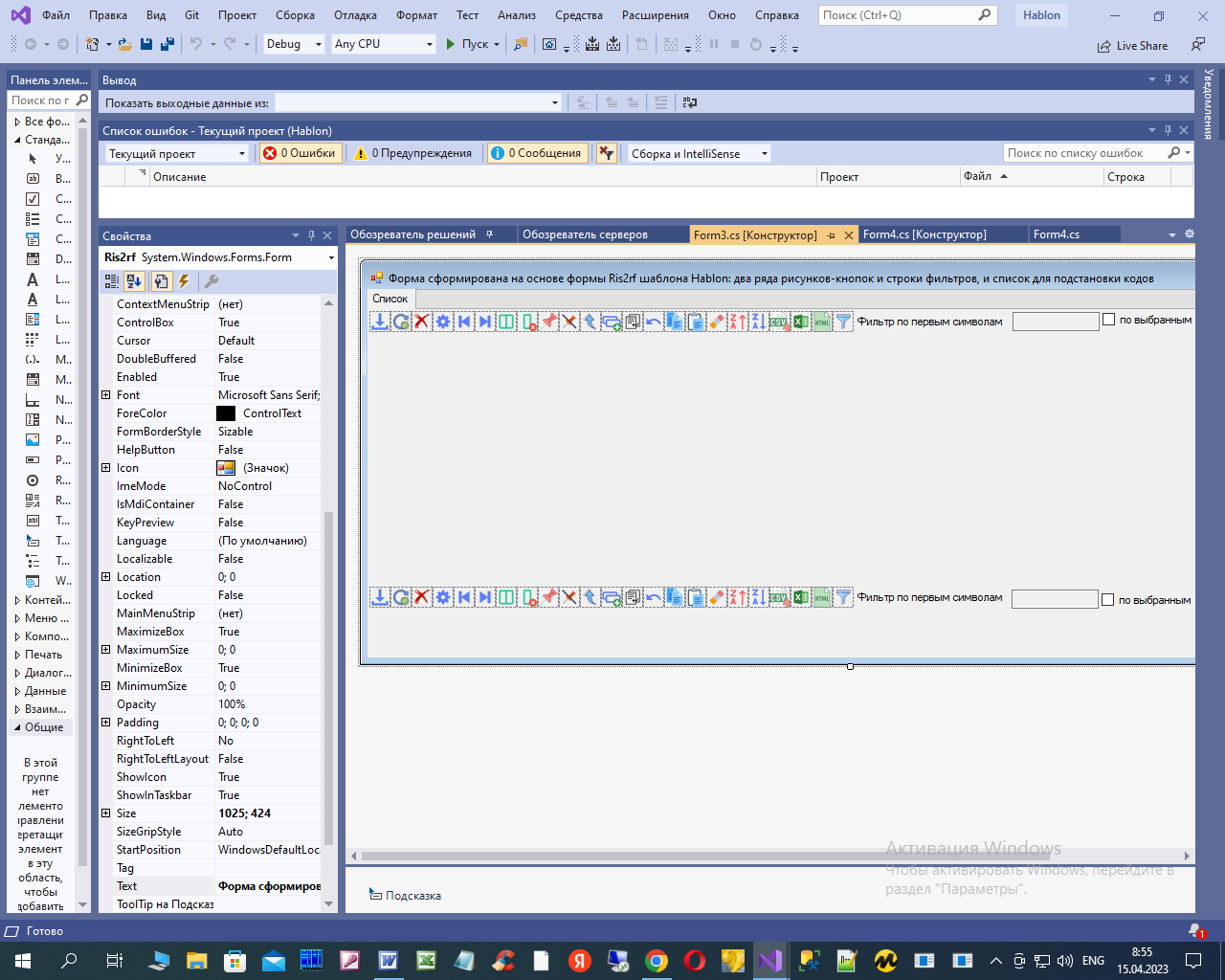
****

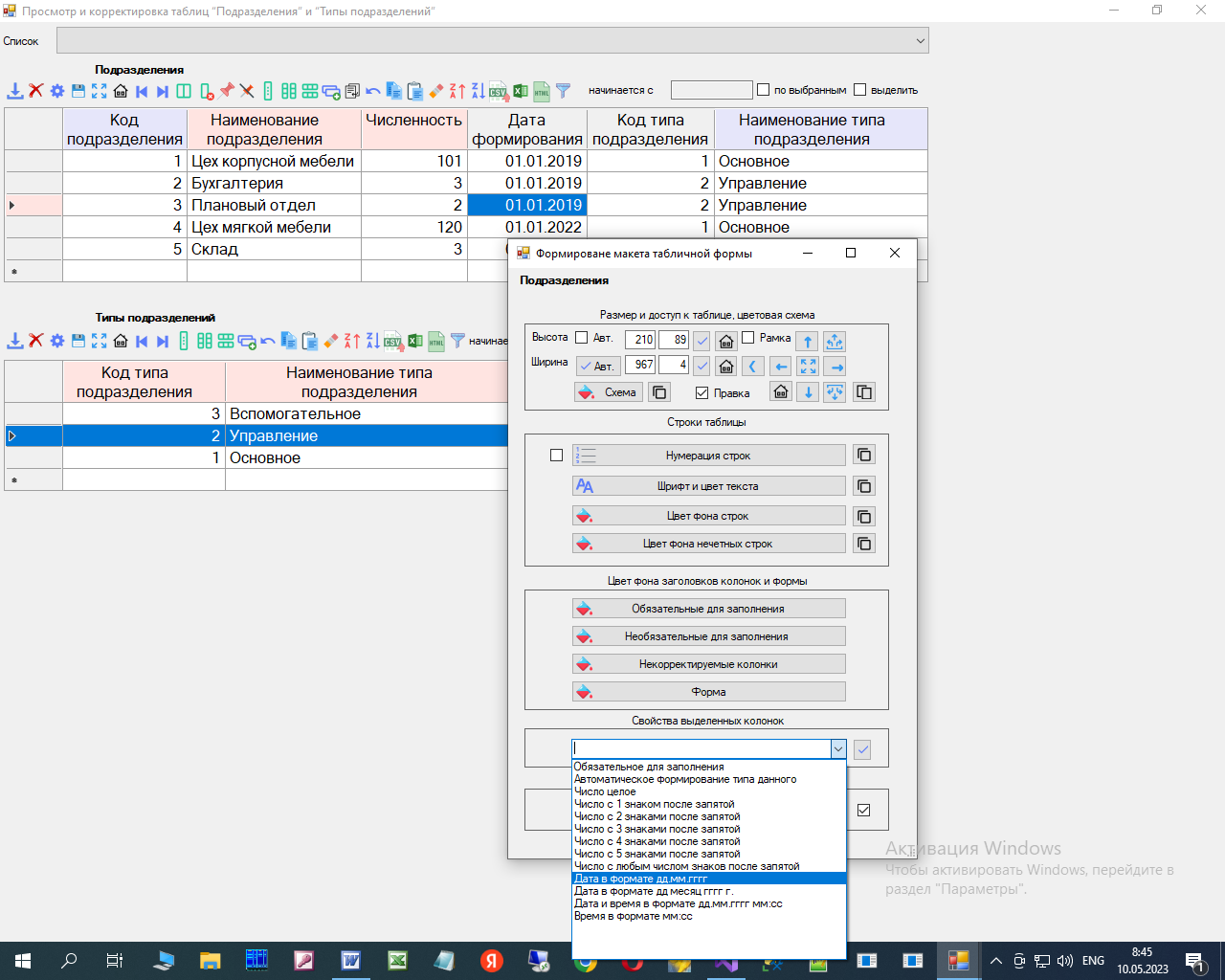
**Рисунок 2 – Просмотр и корректировка таблицы.**

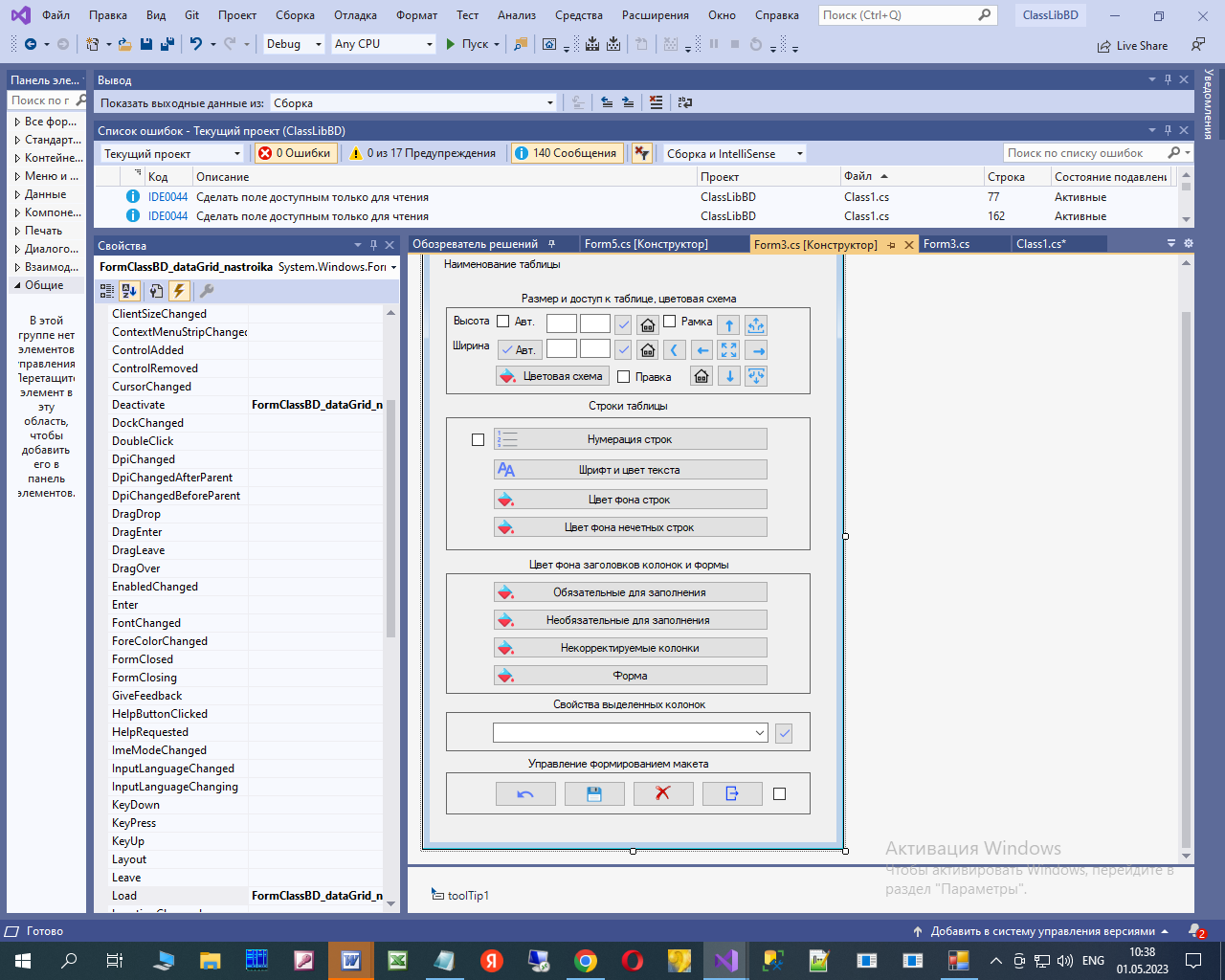
 Загрузить таблицу базы данных в таблицу *dataGridView.*

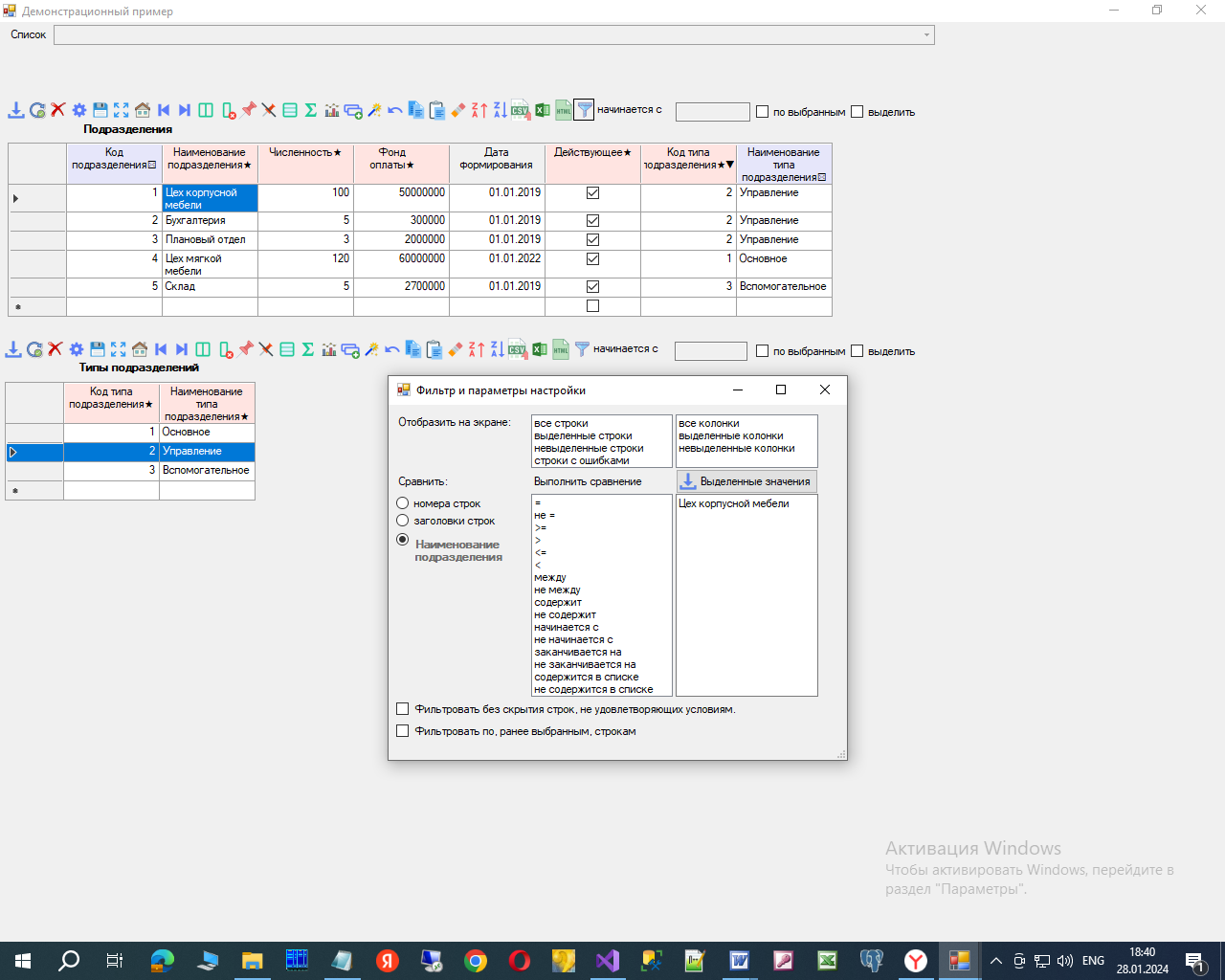
 Обновить базу данных значениями из выделенных строк в таблице.

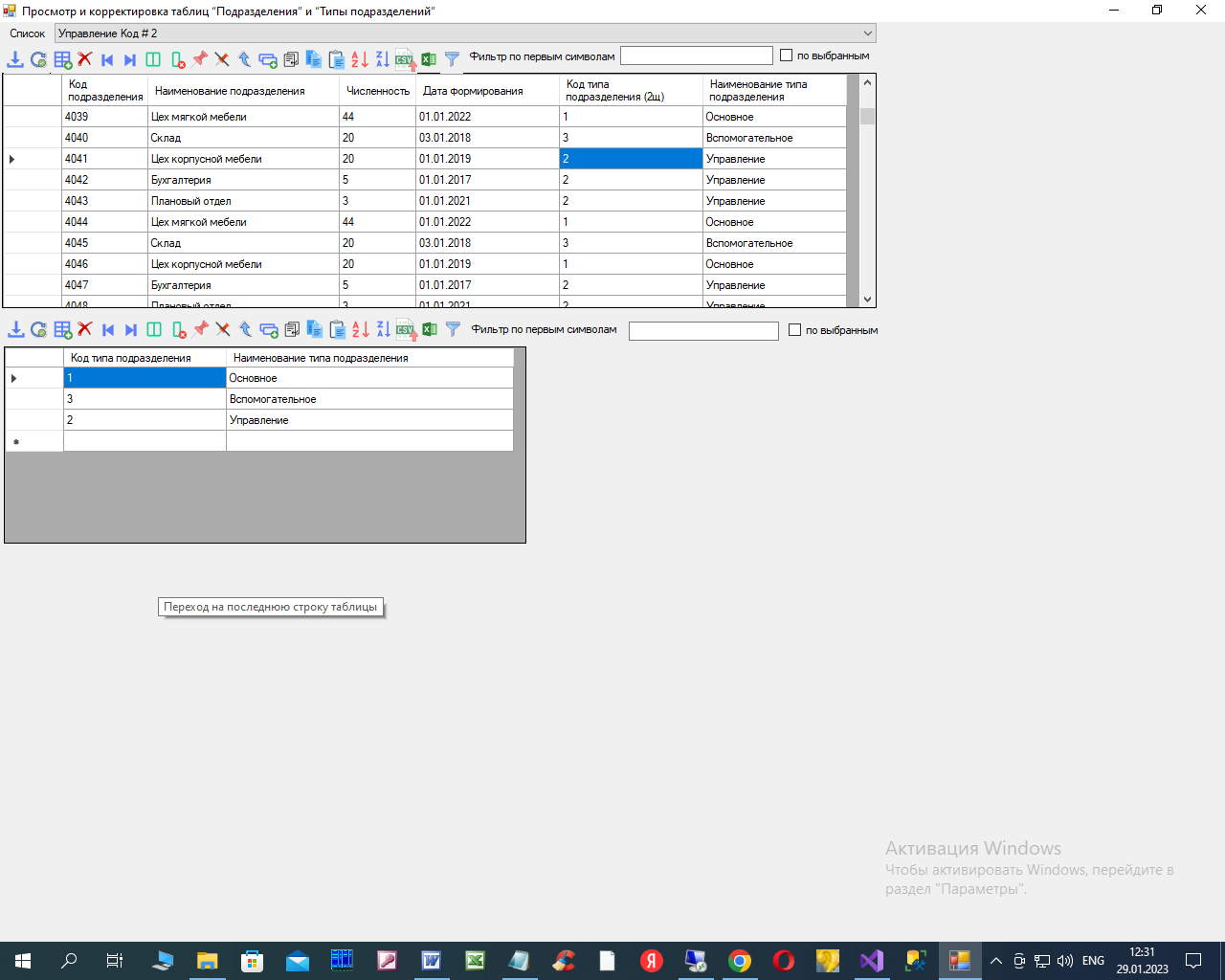
 Удалить из базы данных и из таблицы выделенные строки.

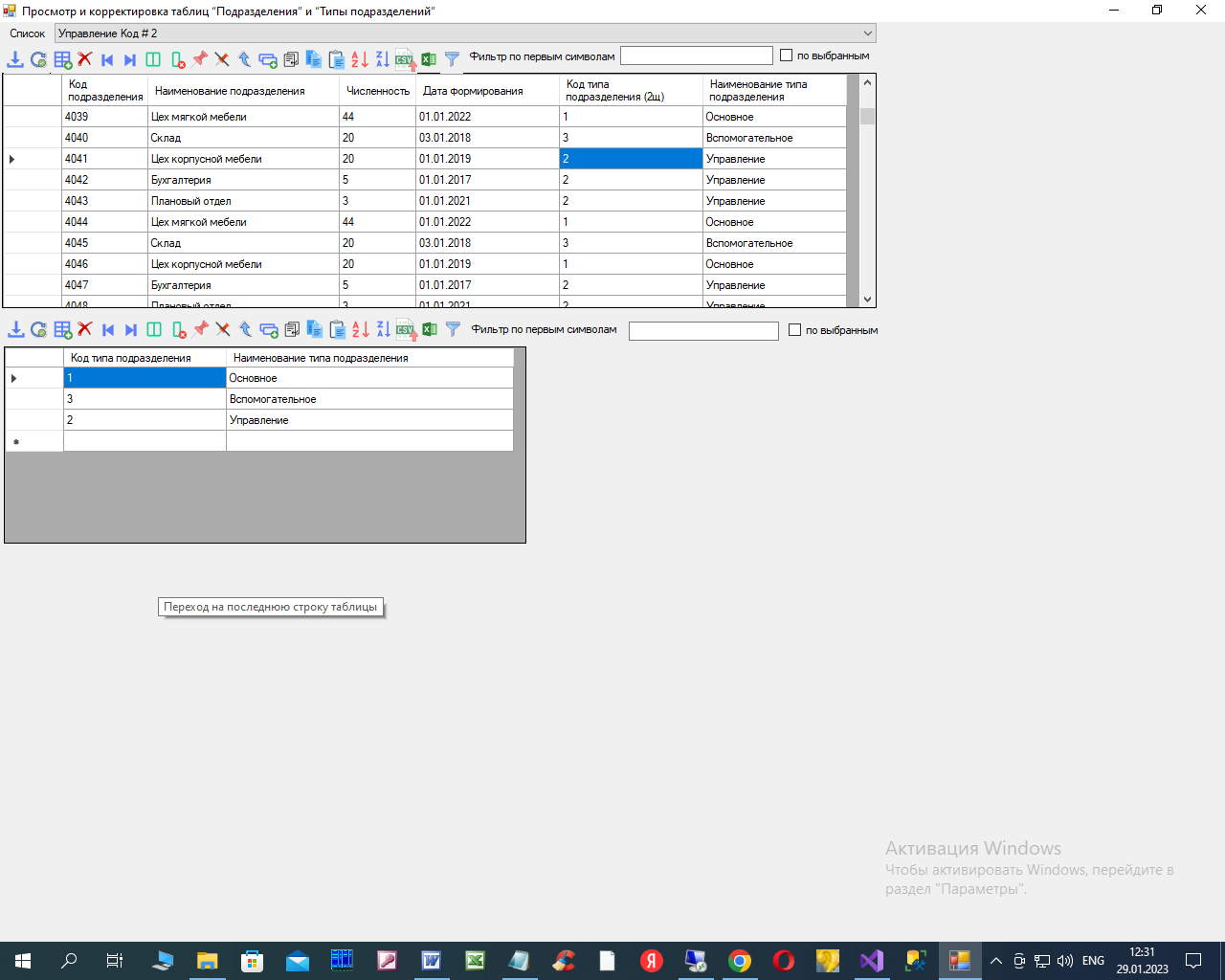
** Конструктор макетов табличной формы с описанием внешнего вида.

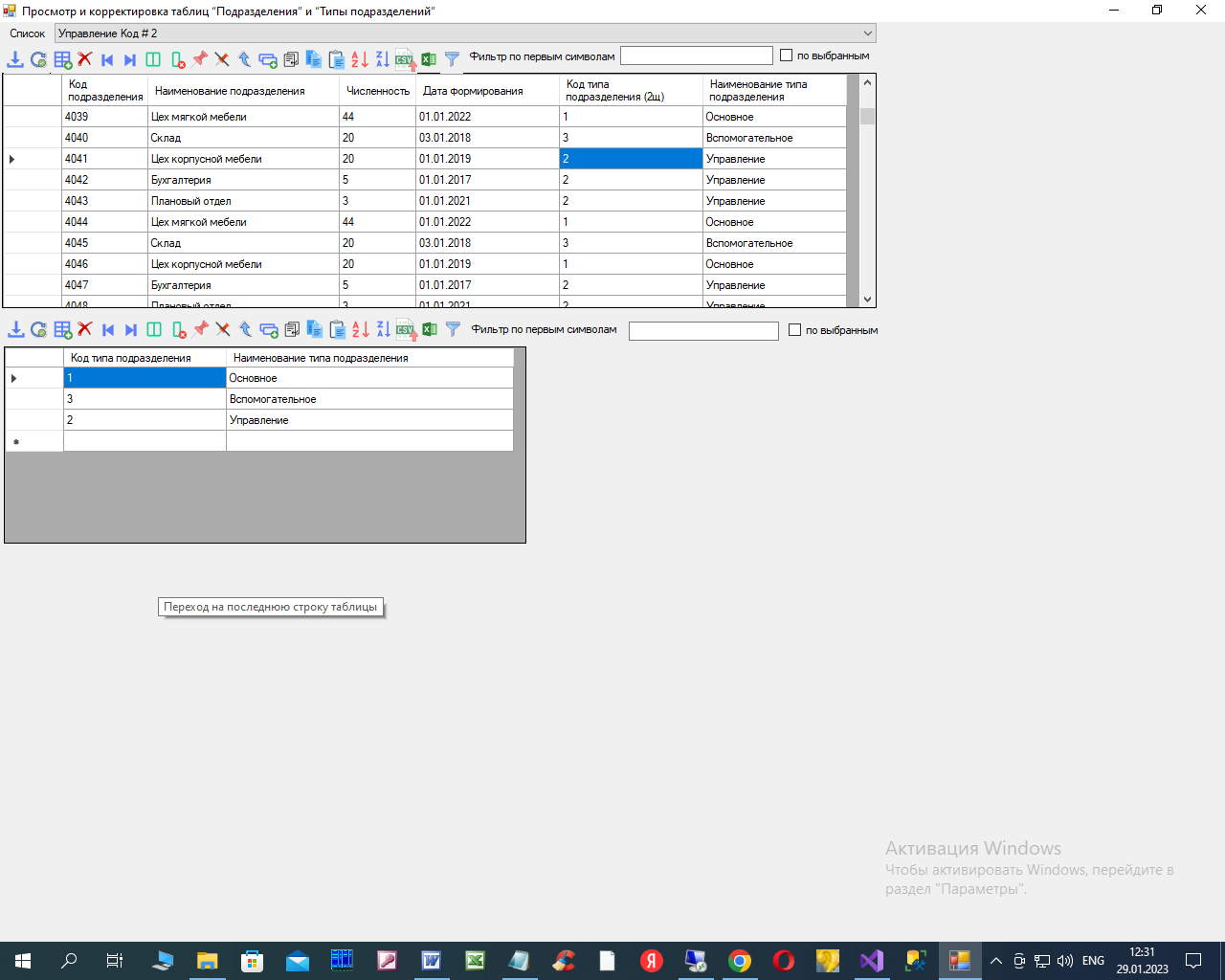
**­** Сохранить макет формы для использования в дальнейшем.

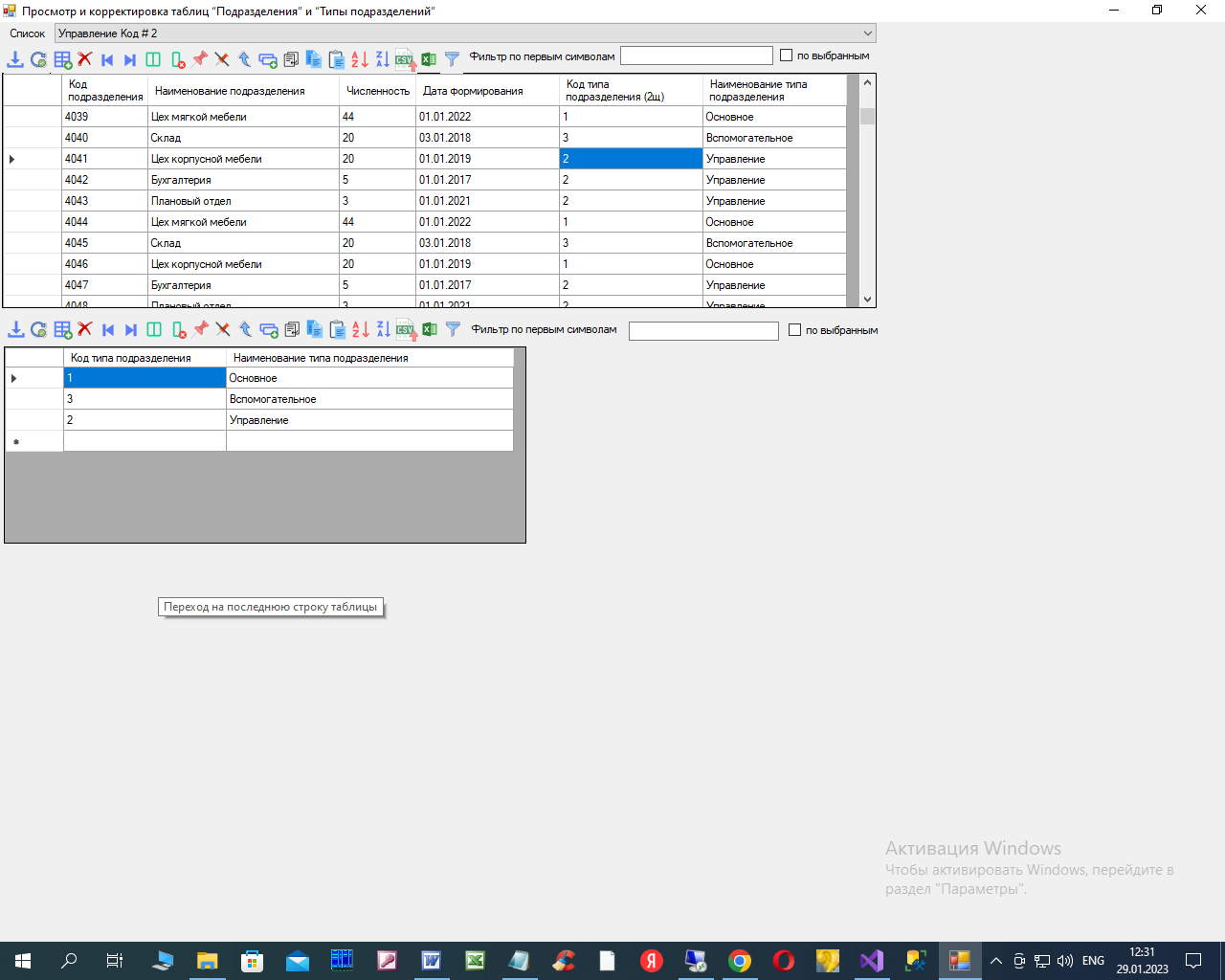
 Развернуть таблицу на весь экран дисплея.

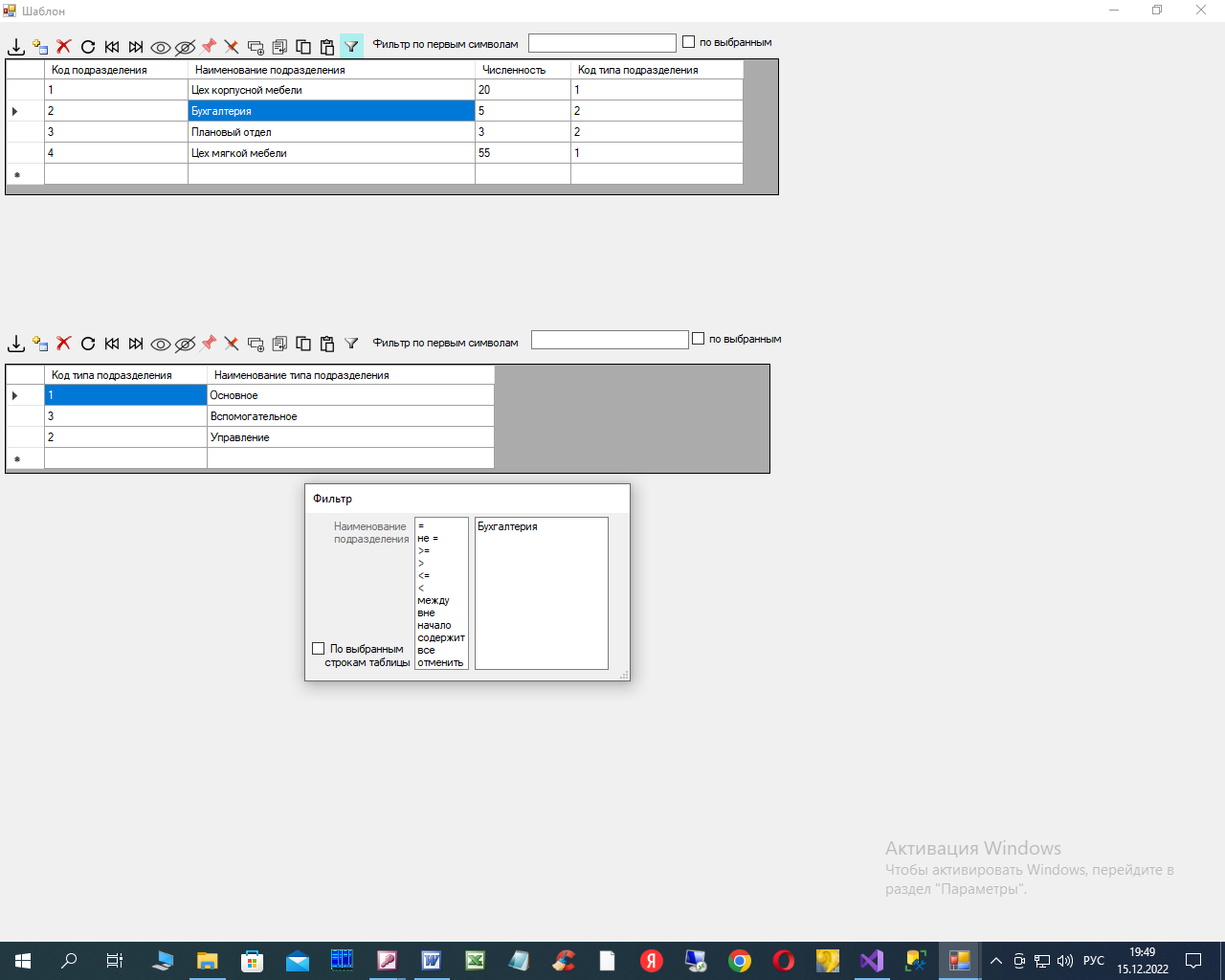
 Свернуть таблицу до первоначального размера.

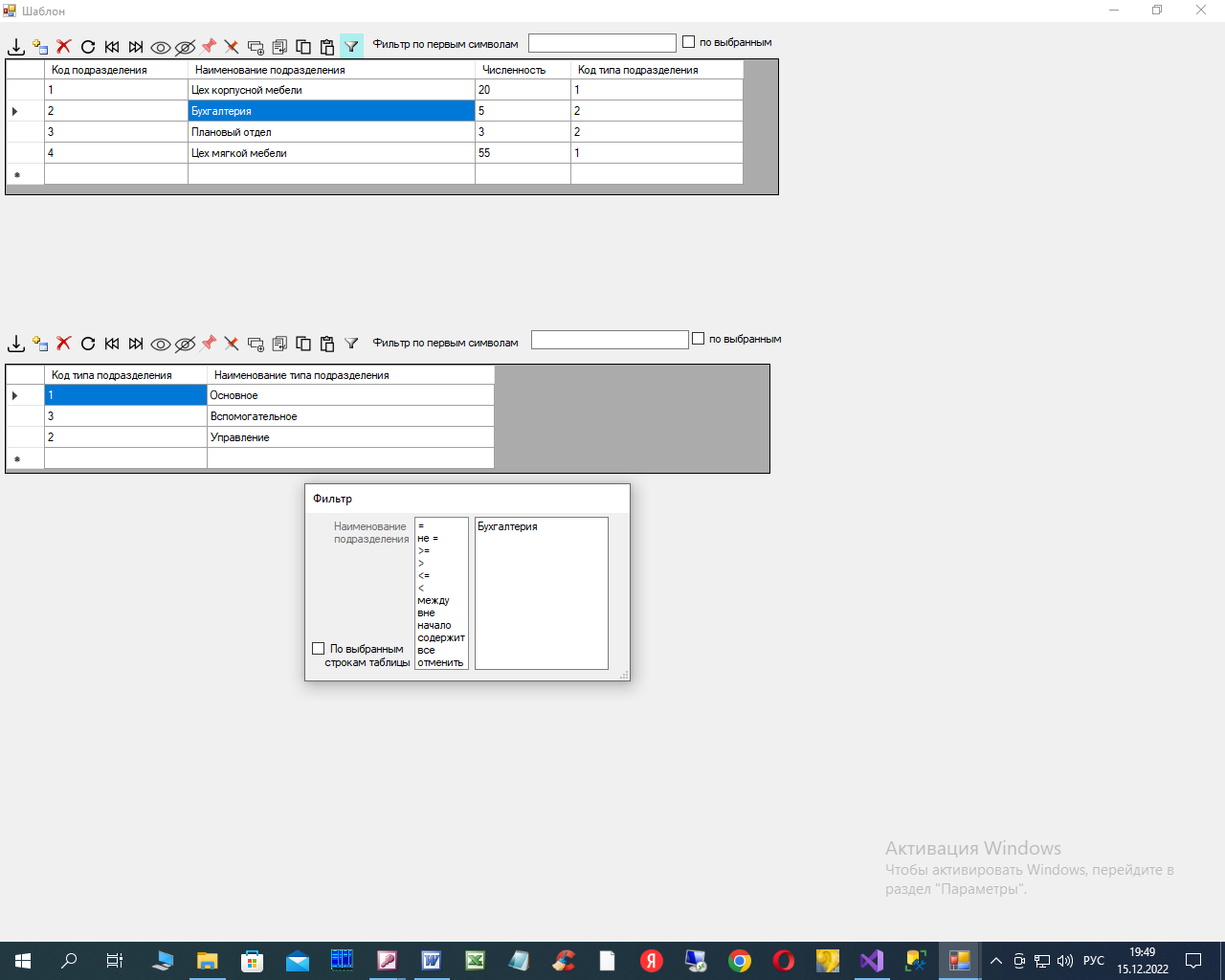
 Перейти на первую видимую строку таблицы с учетом фильтра.

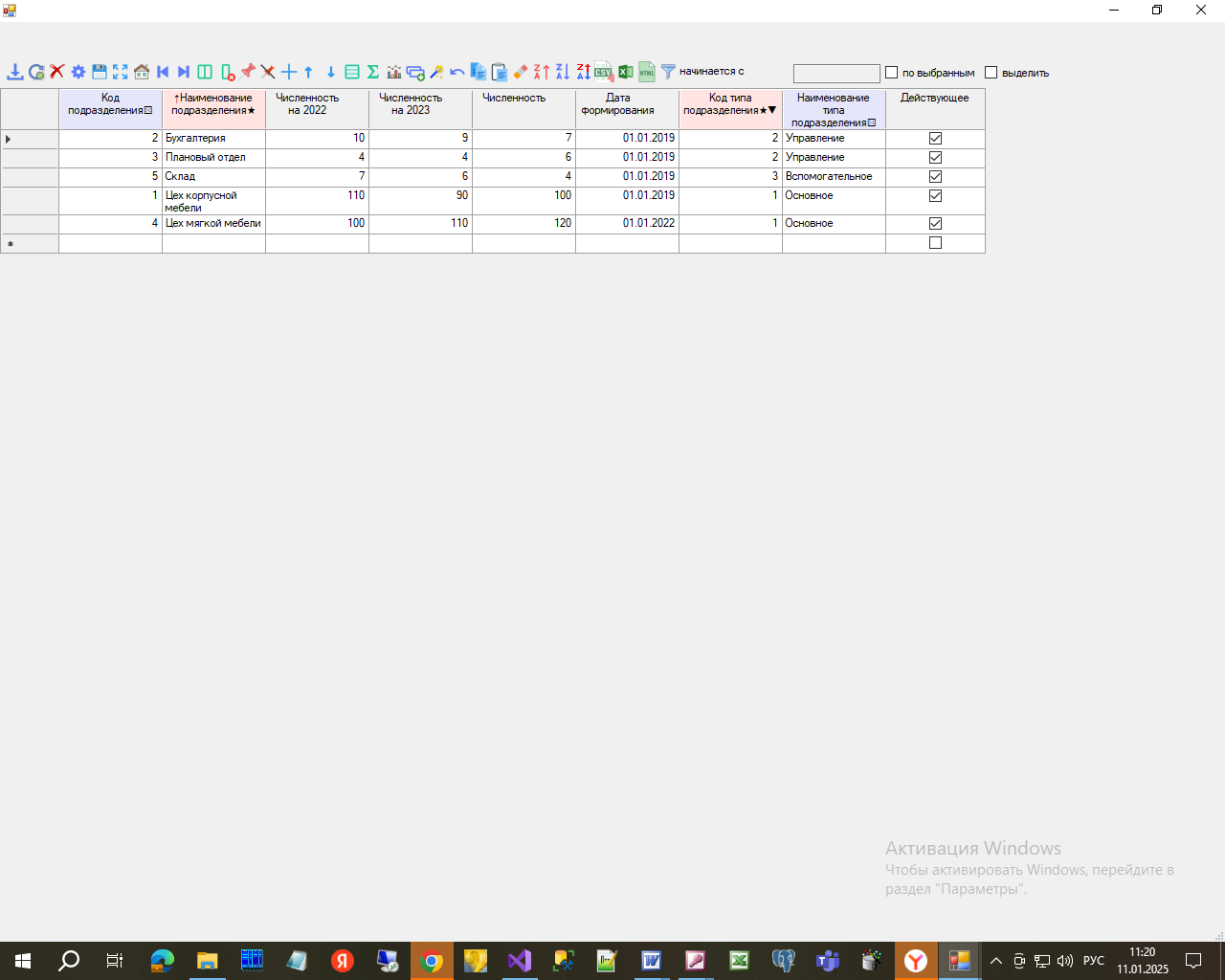
 Перейти на последнюю видимую строку таблицы с учетом фильтра.

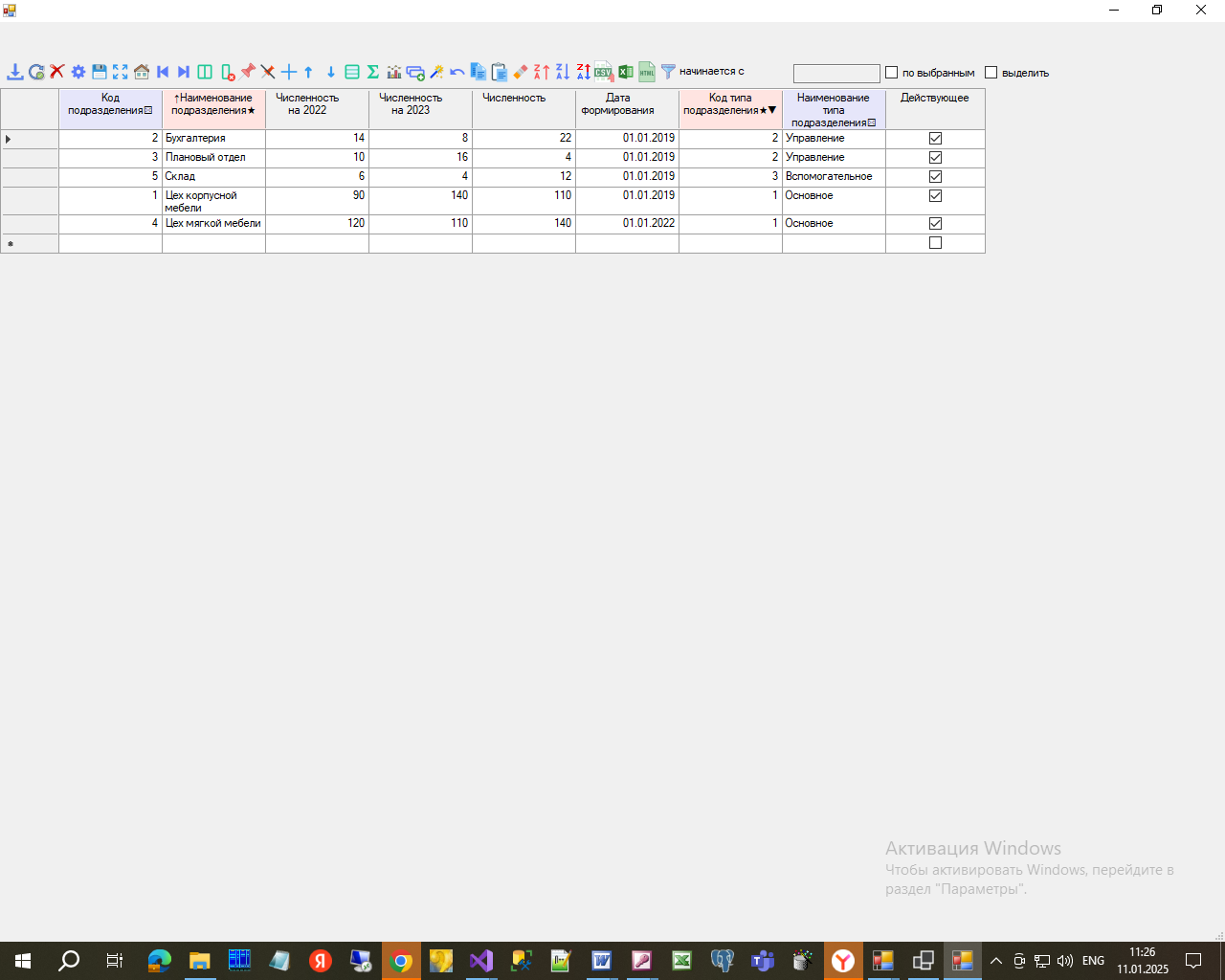
 Показать все колонки таблицы или только выделенные.

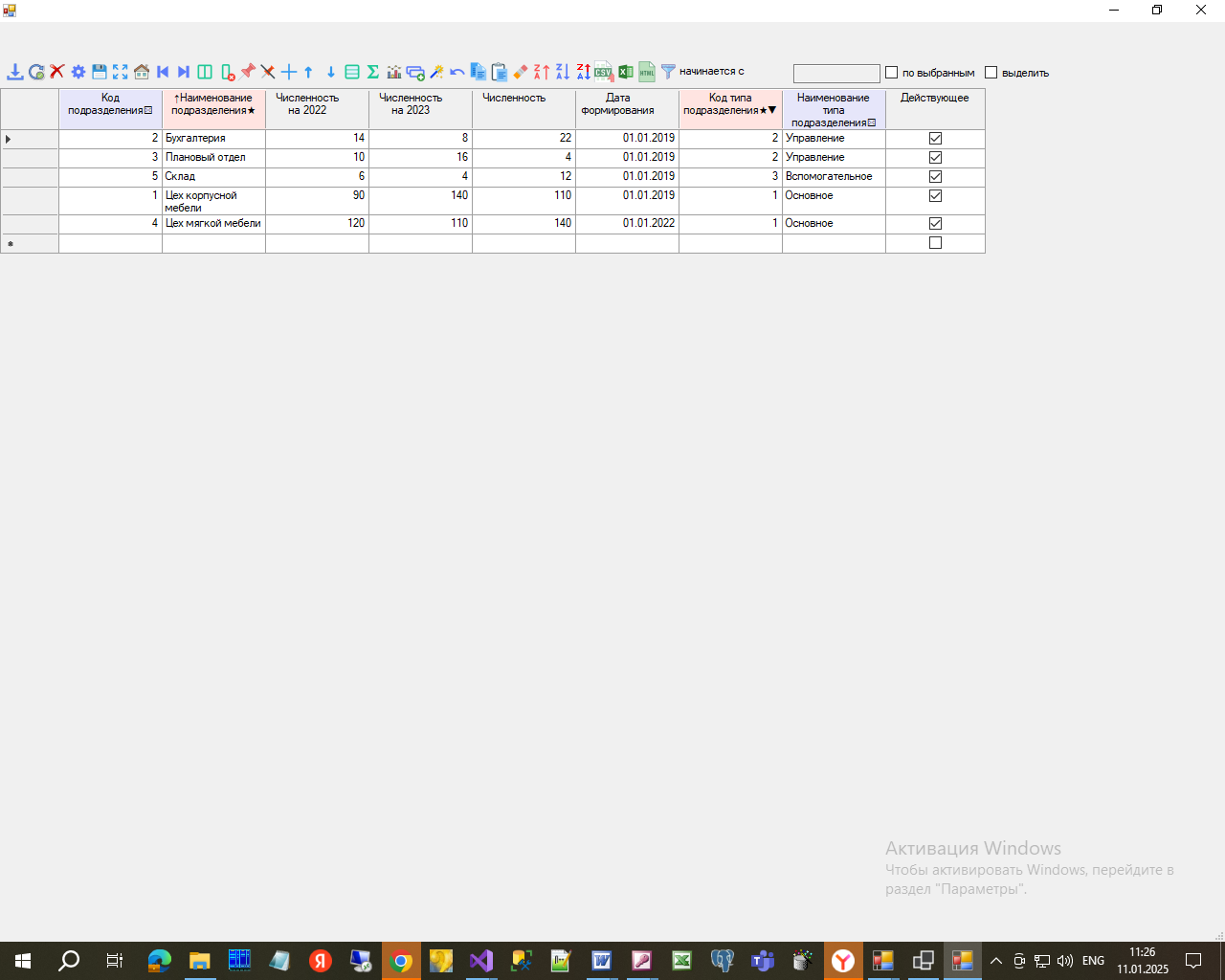
 Скрыть выделенные колонки таблицы.

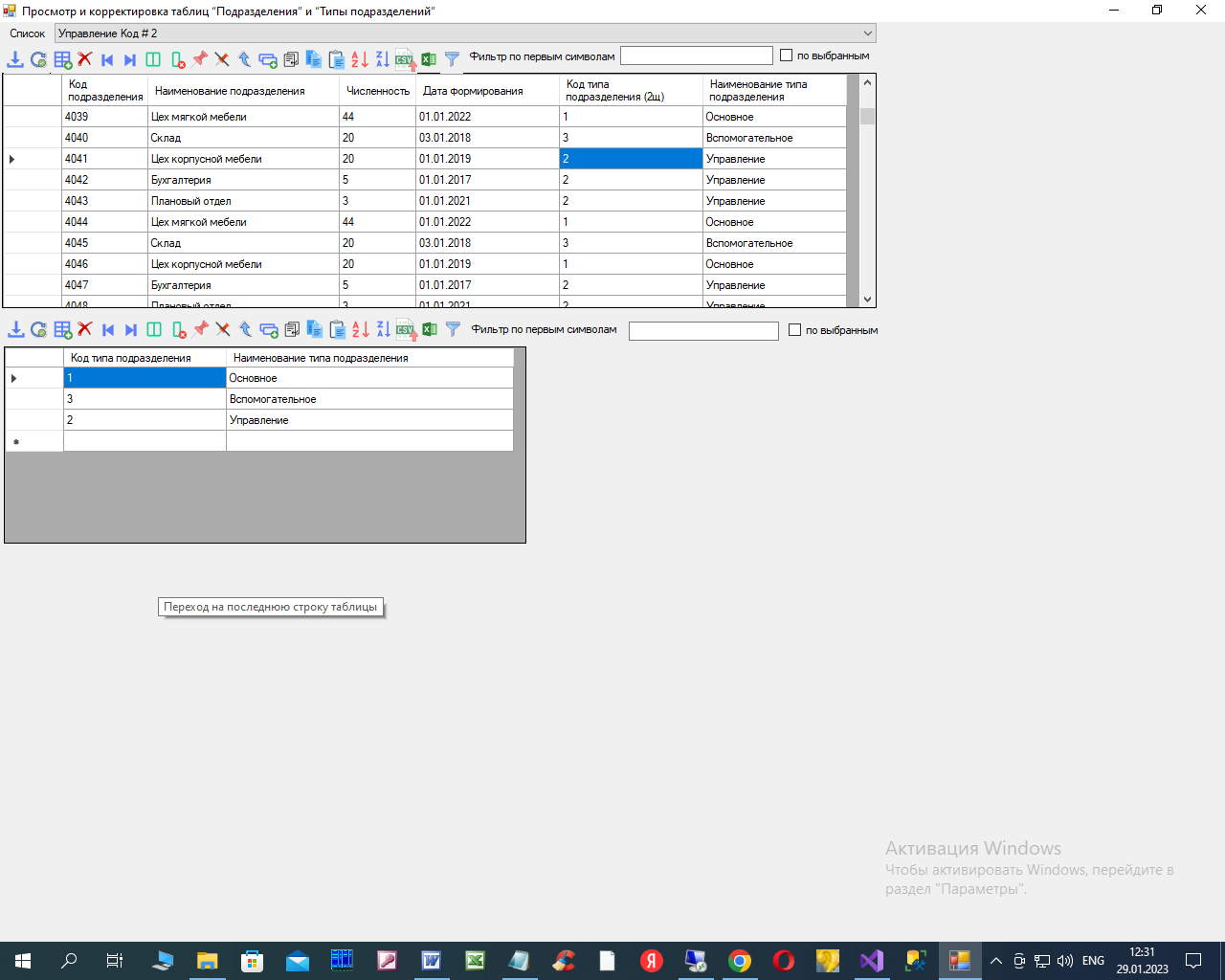
 Закрепить текущую колонку таблицы и все колонки слева от её.

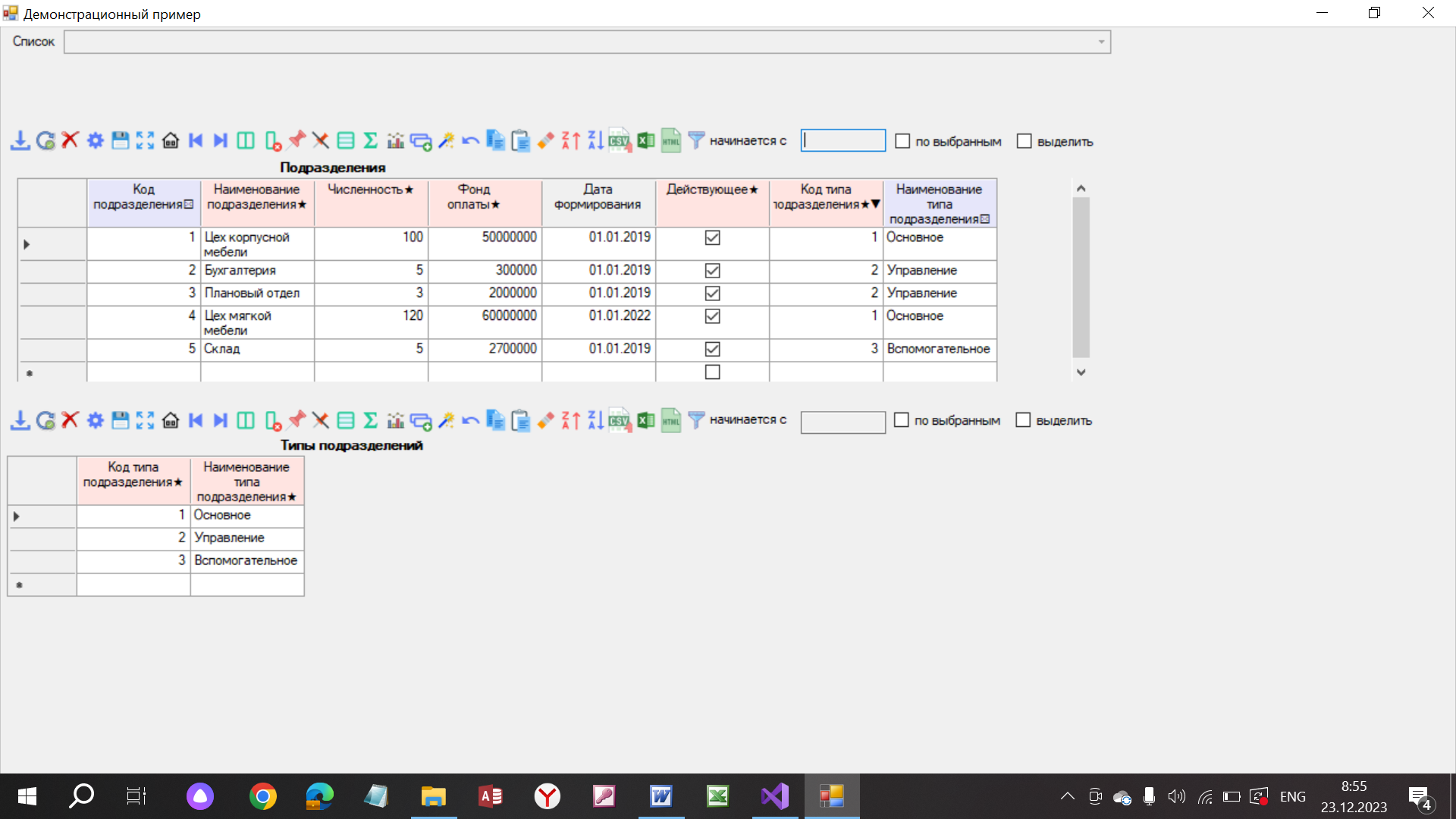
 Освободить все колонки от закрепления и удалить признаки сортировки (вертикальные стрелочки) в заголовках выделенных колонок сортировки.

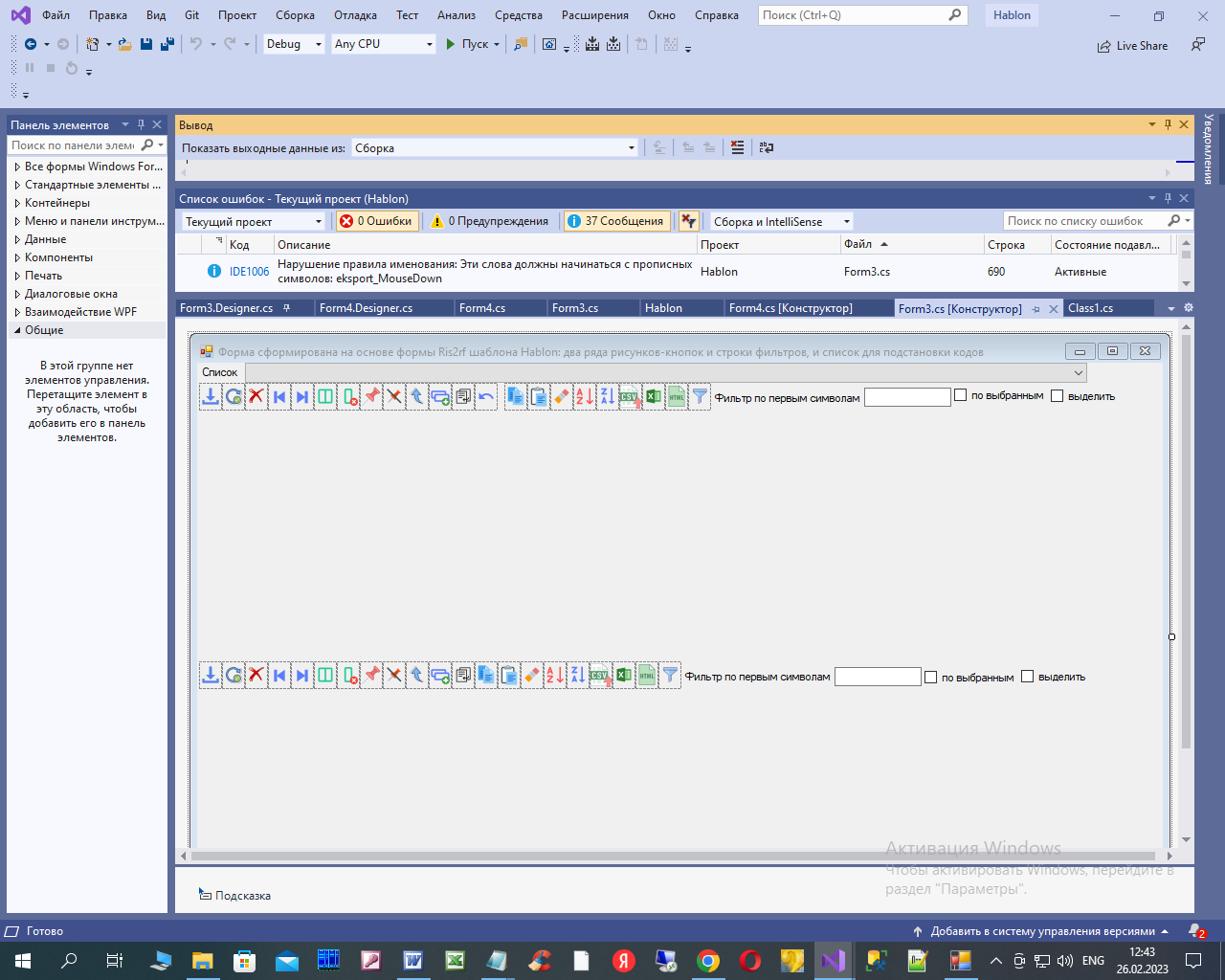
**** Добавить в таблицу пустую строку после выделенной строки.

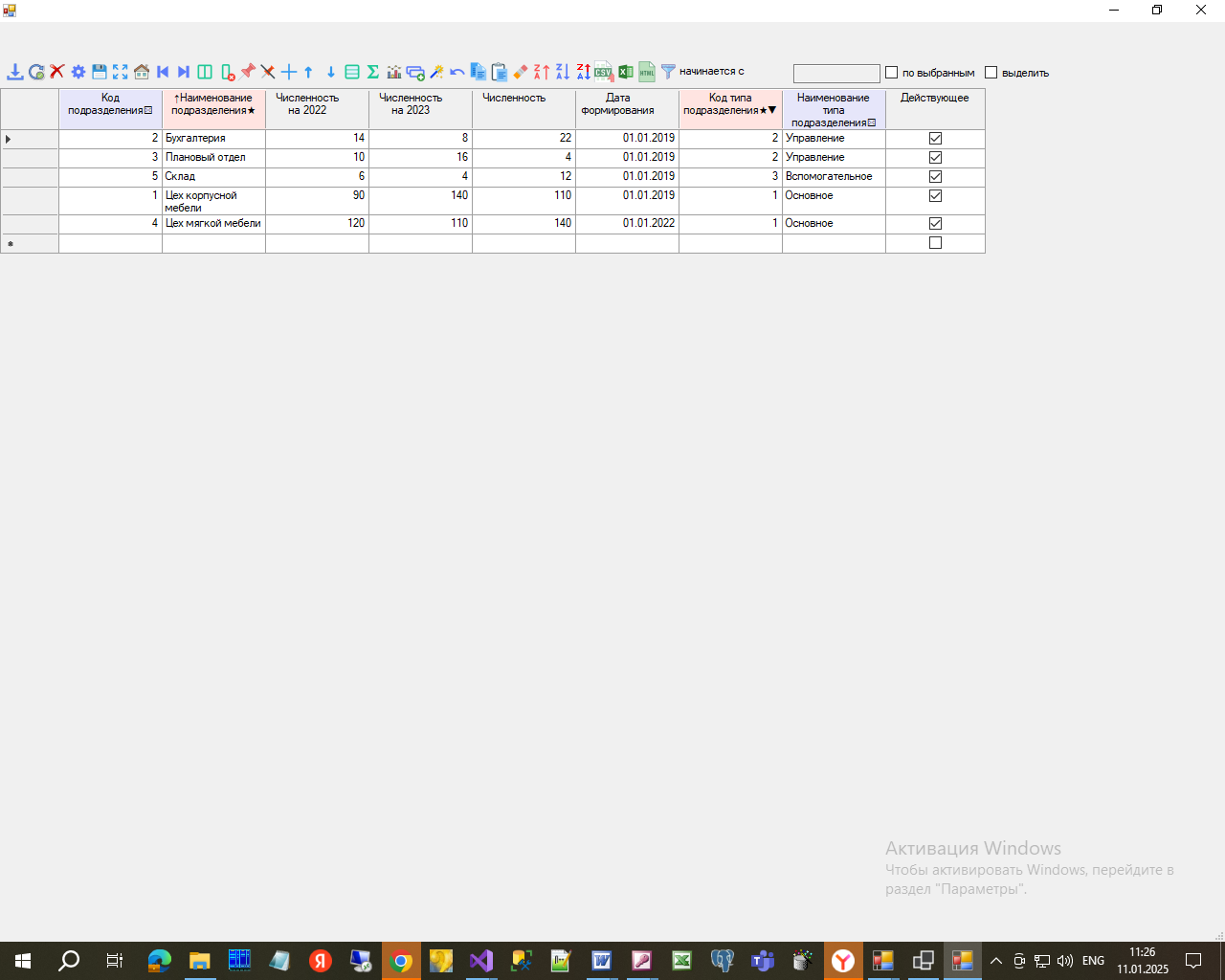
****Переместить выделенные строки на одну строку вверх или вниз.

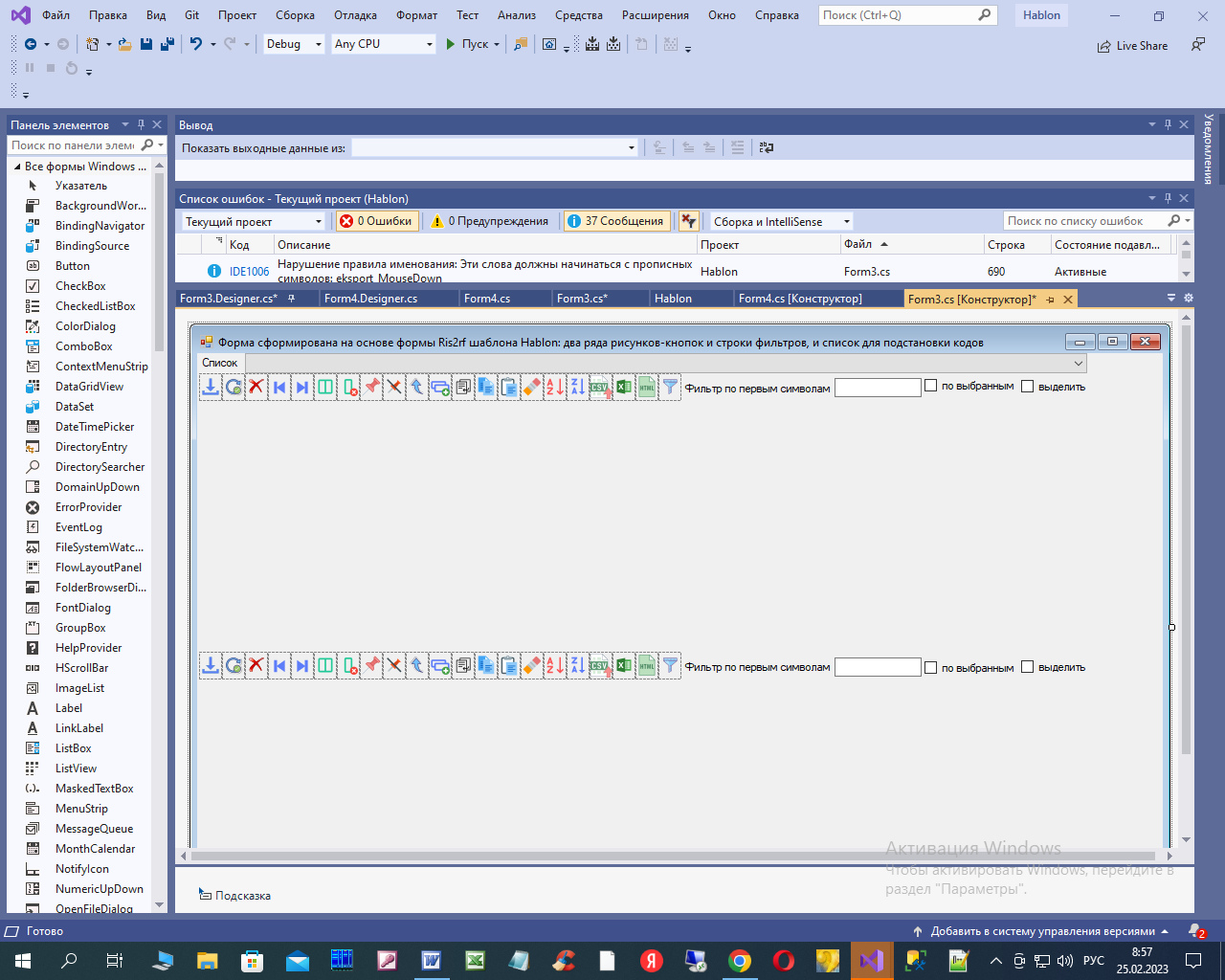
****Конструкторы ленточных форм, итоговых таблиц и диаграмм.

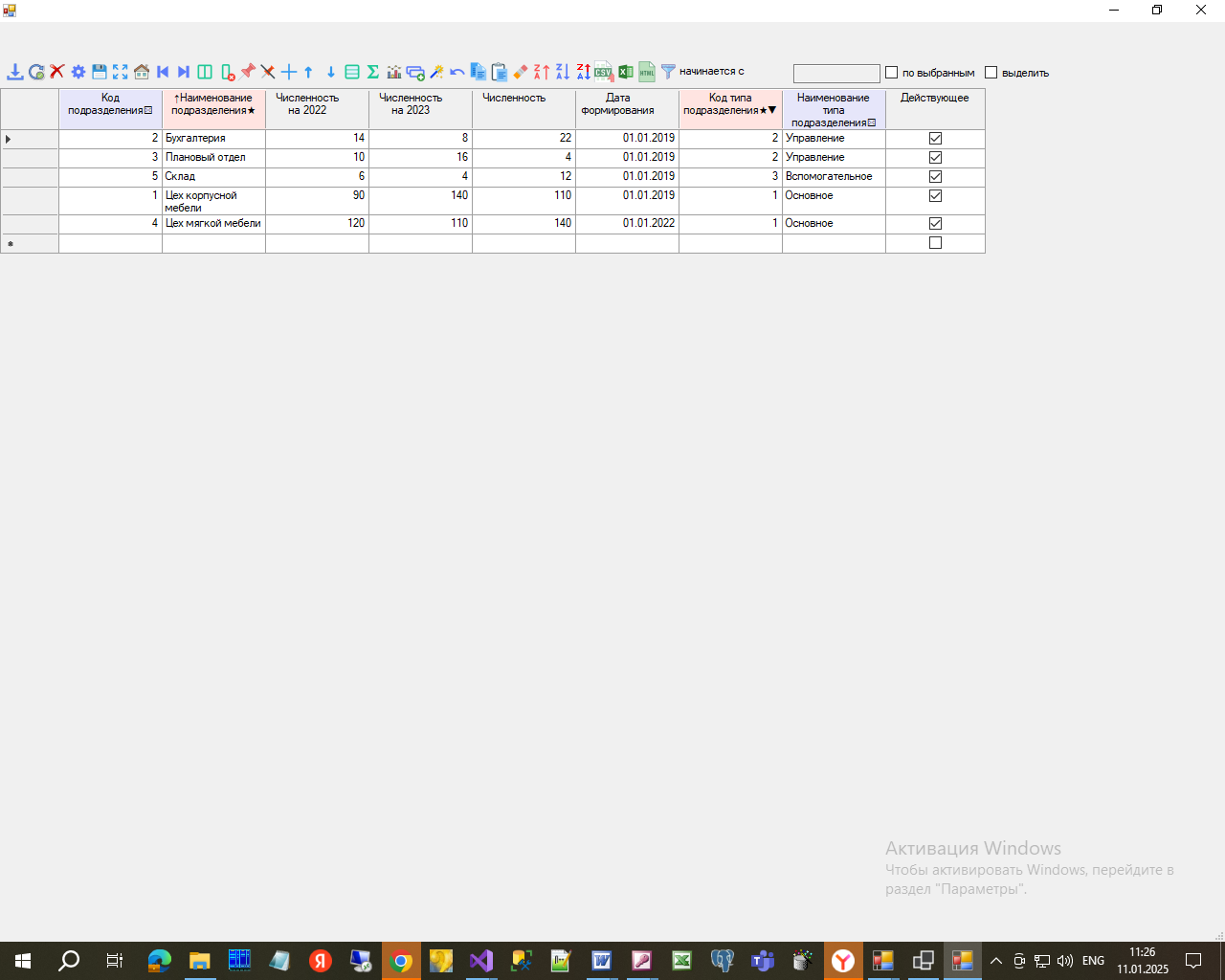
Копировать в конец таблицы выделенные строки.

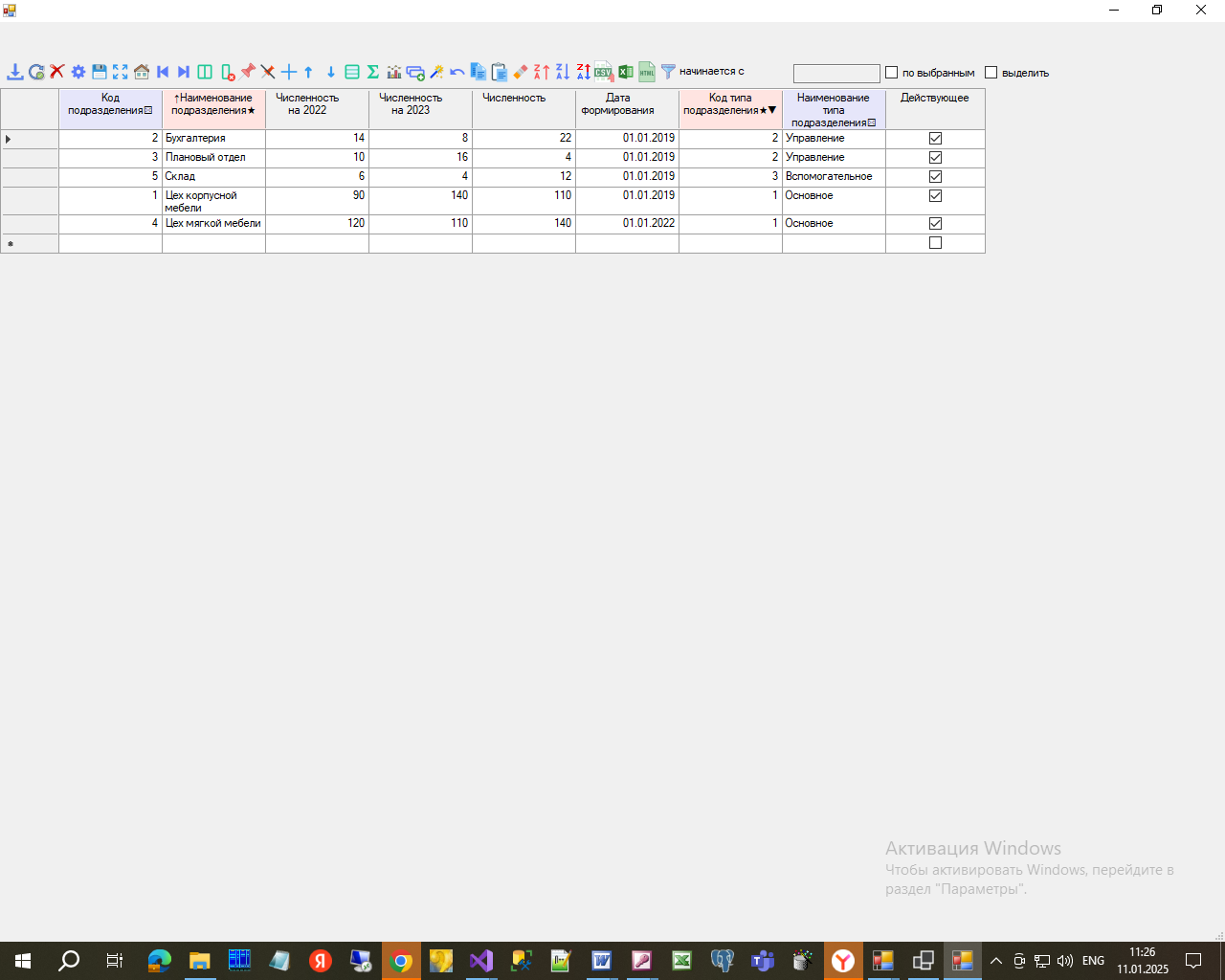
 Конструктор значений, выражений, изображений, расчетных колонок и файлов, связанных с БД..

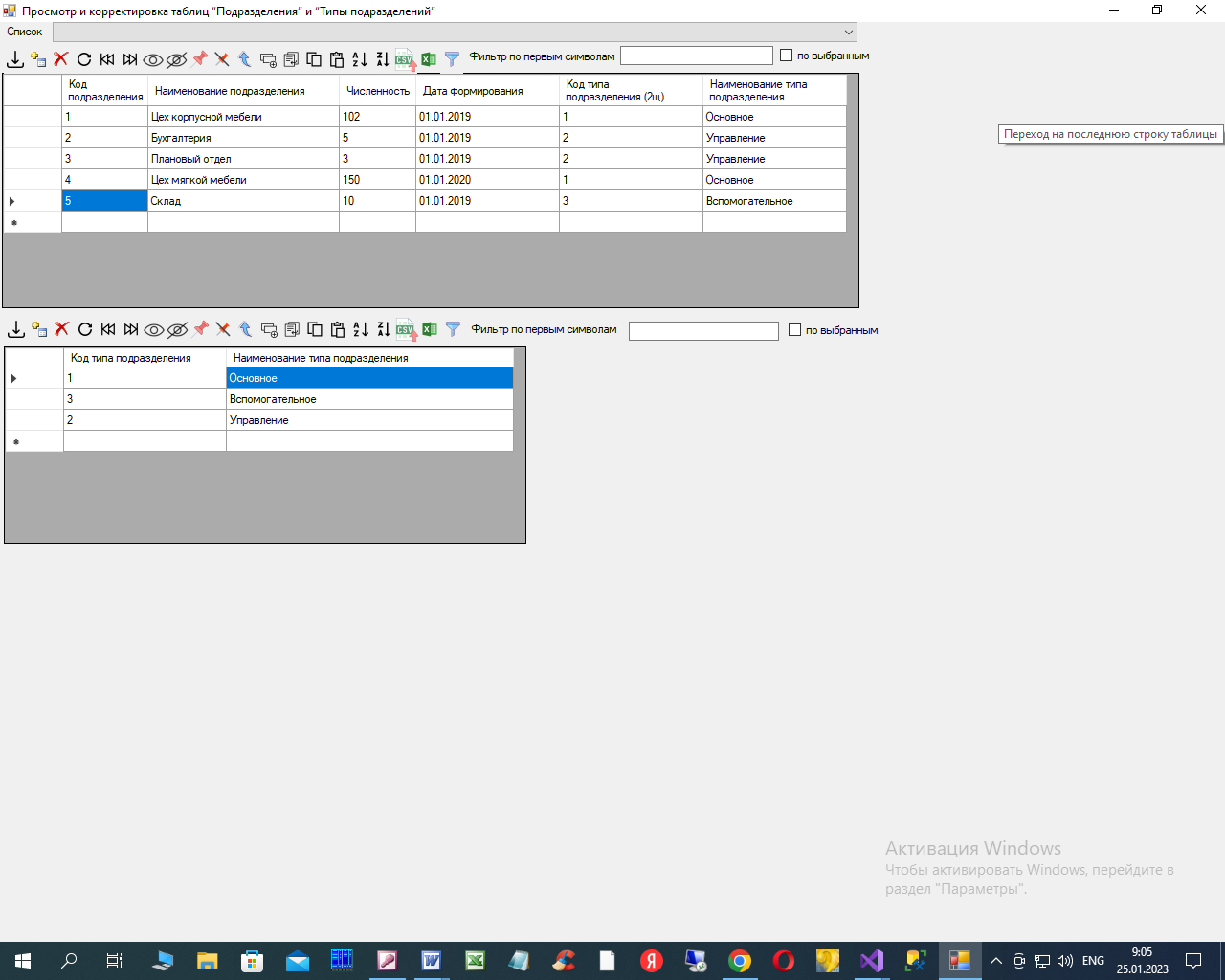
******Отменить последнее изменение в базе данных.

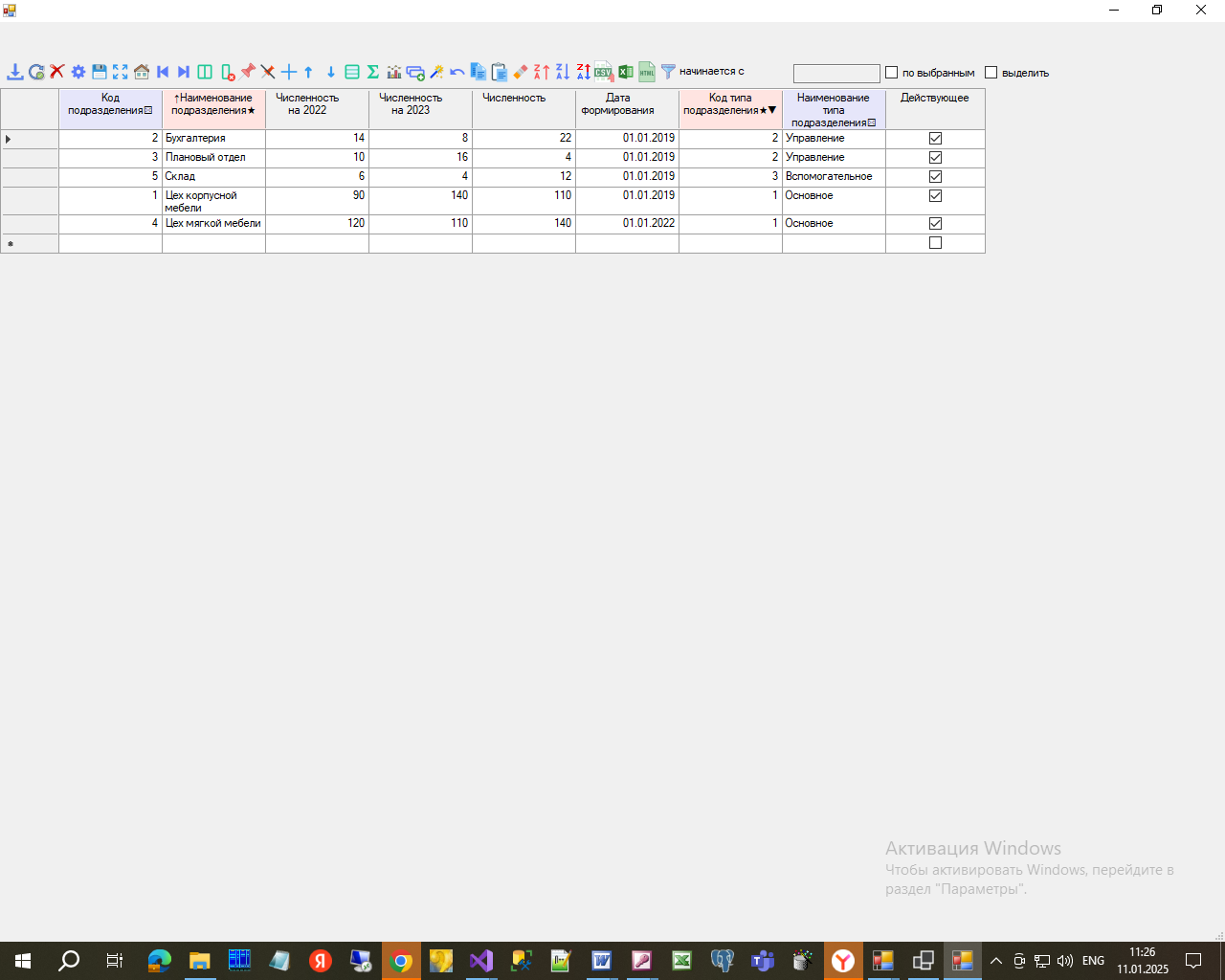
**** Копировать выделенные ячейки в буфер обмена и вставить из буфера.

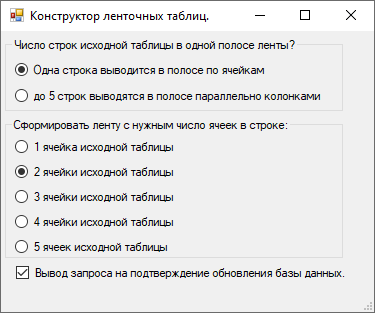
 Очистить выделенные ячейки в таблице.

**** Сортировать по всем выделенным колонкам по возрастанию или убыванию или по отдельным колонкам в различном порядке.

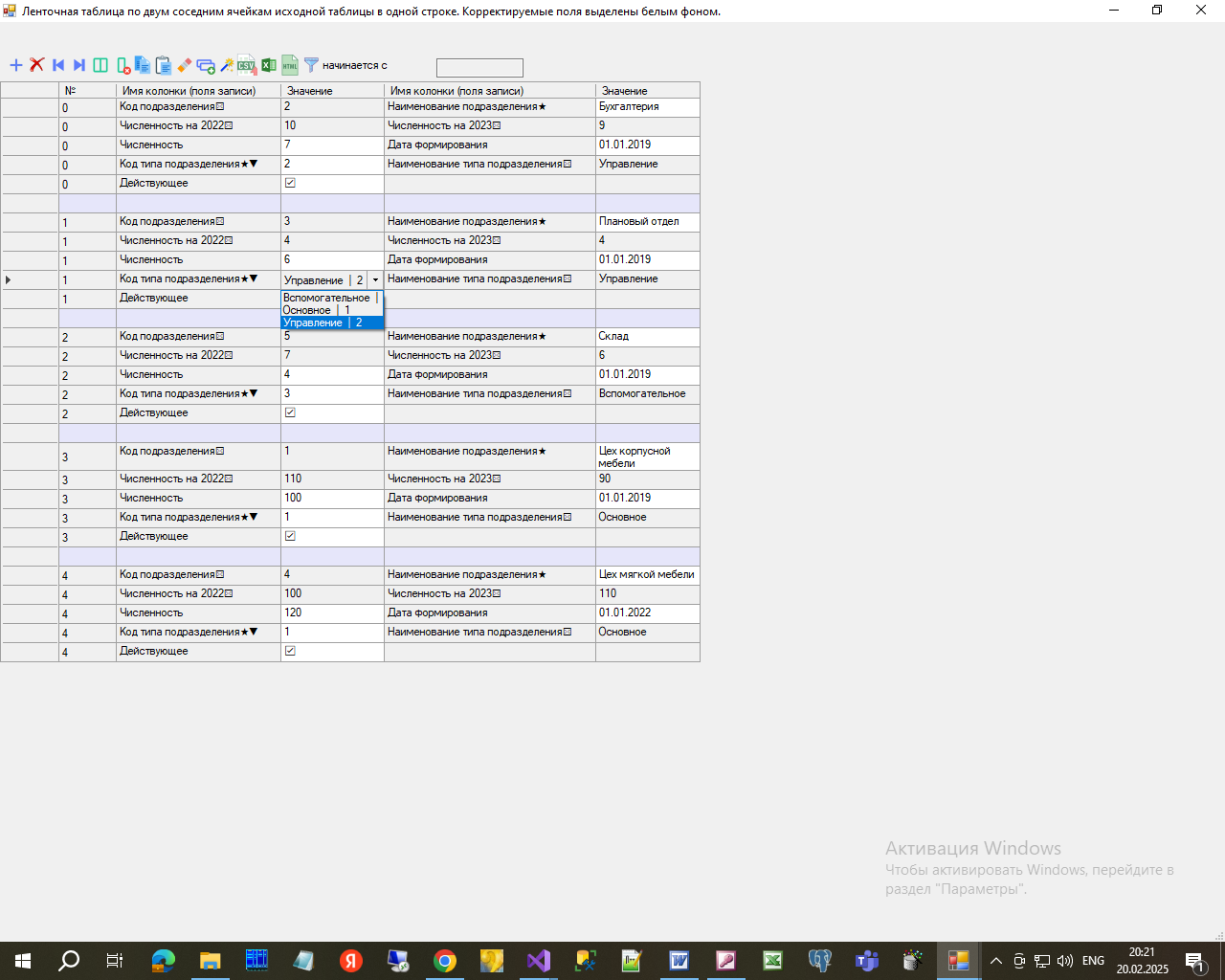
**** Экспорт выделенных строк в *CSV*, *Excel*, *HTML* – файлы.

Конструктор фильтров.

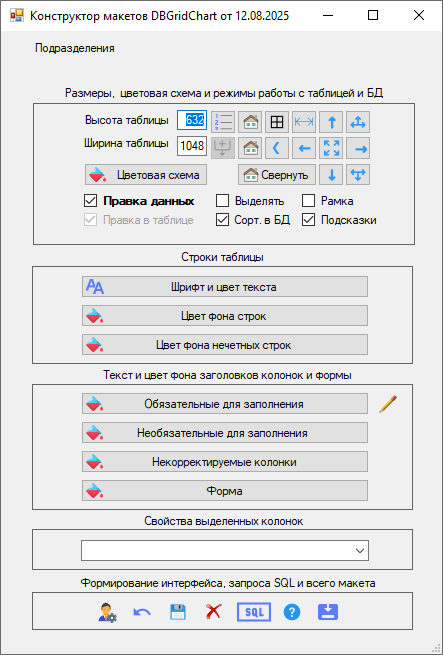
**** Оперативный фильтр в строке интерфейса.

****

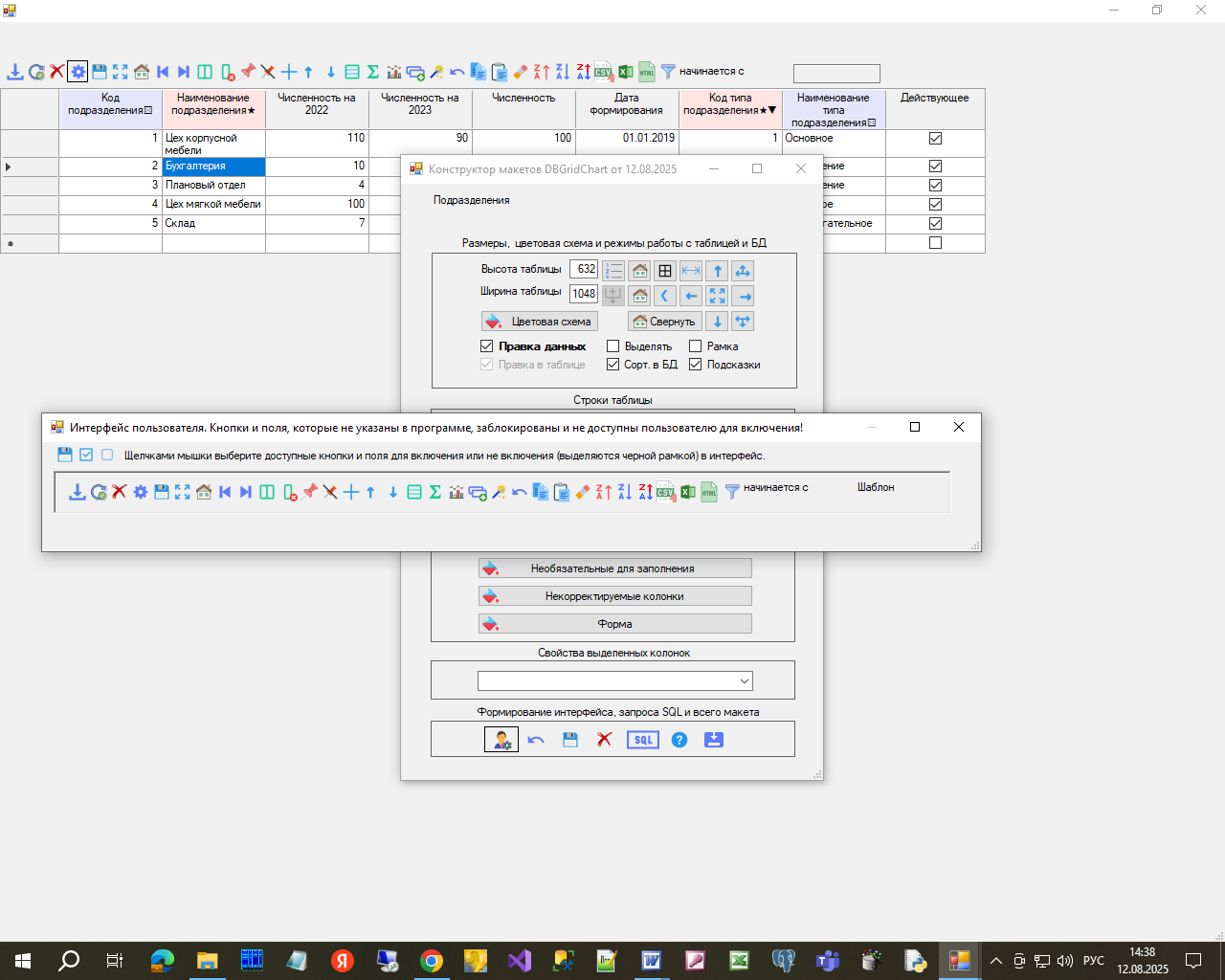
**Рисунок 3– Конструктор ленточных форм.**

****

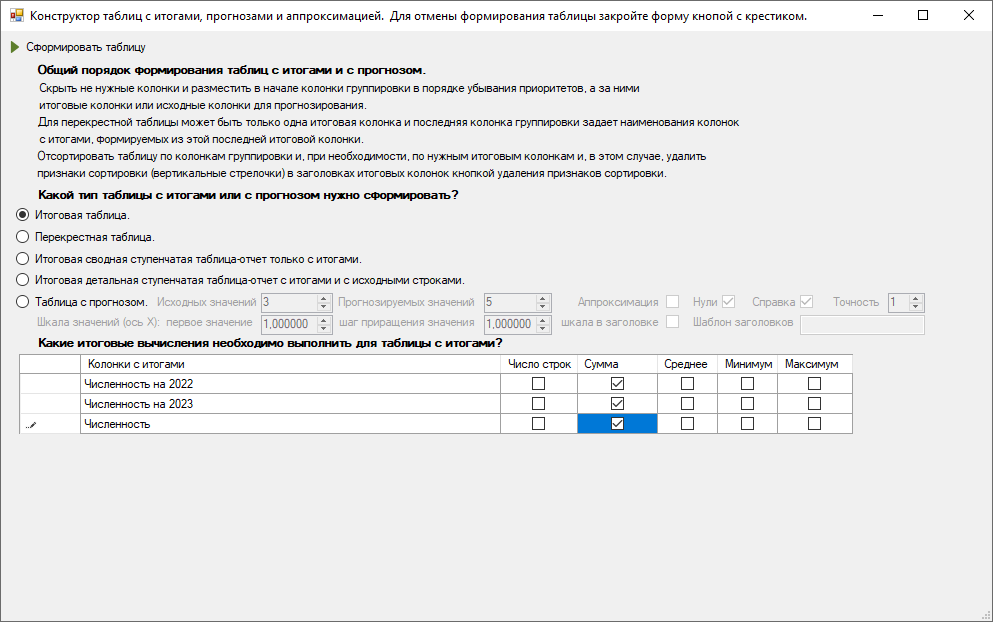
**Рисунок 4 – Ленточная форма.**

****

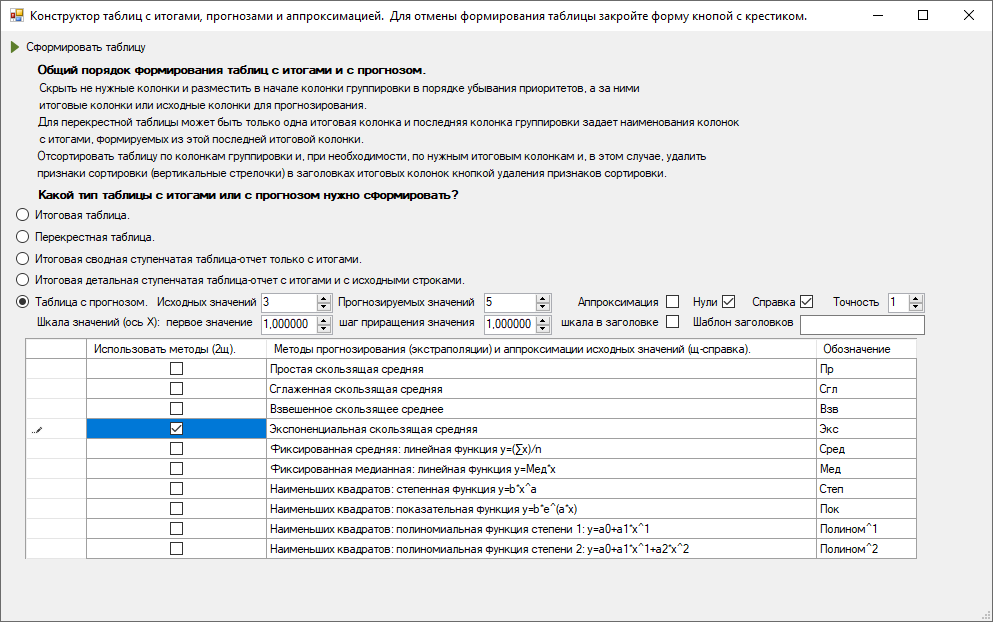
**Рисунок 5 – Конструктор макетов.**

****

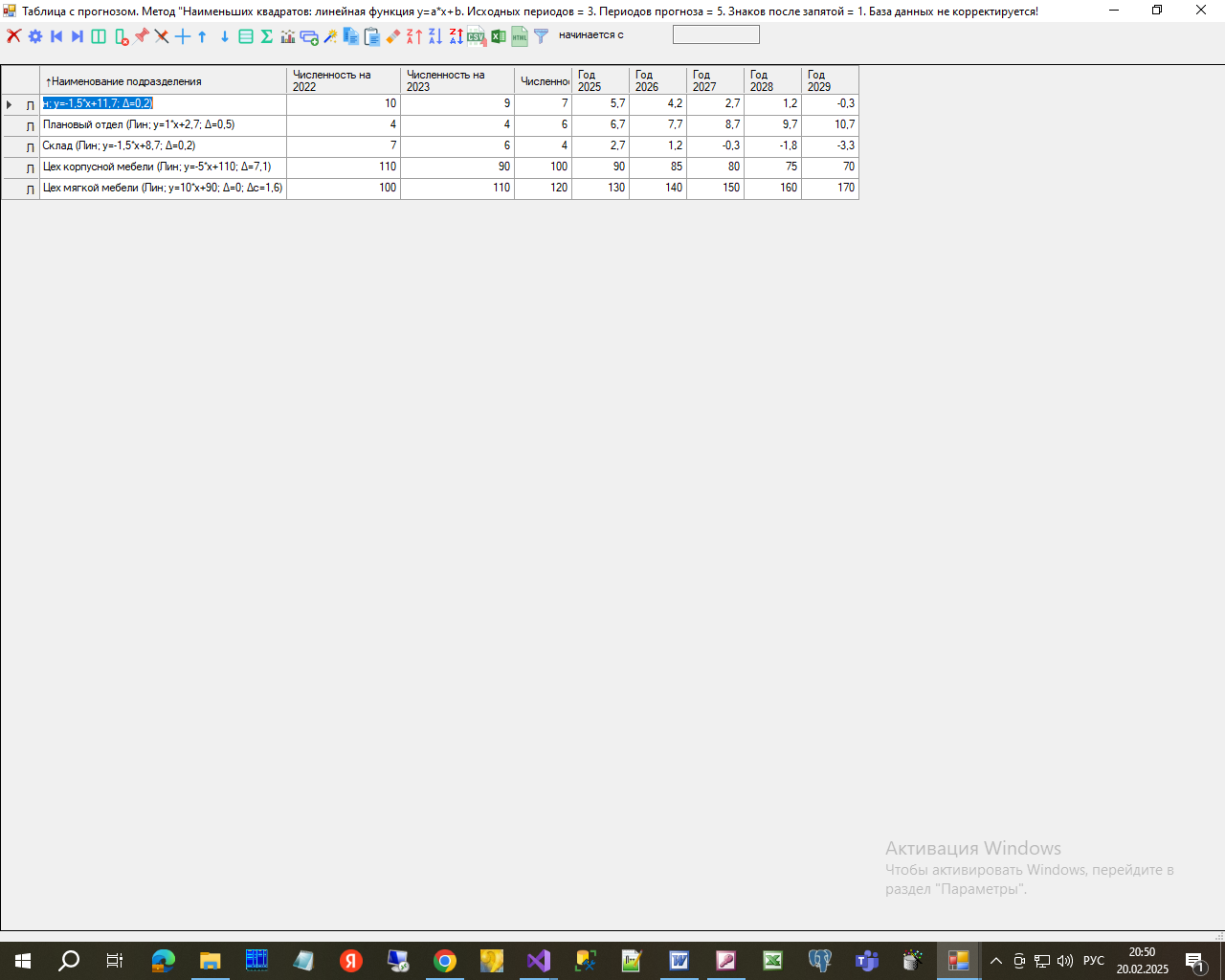
**Рисунок 6 – Конструктор интерфейса пользователя**

****

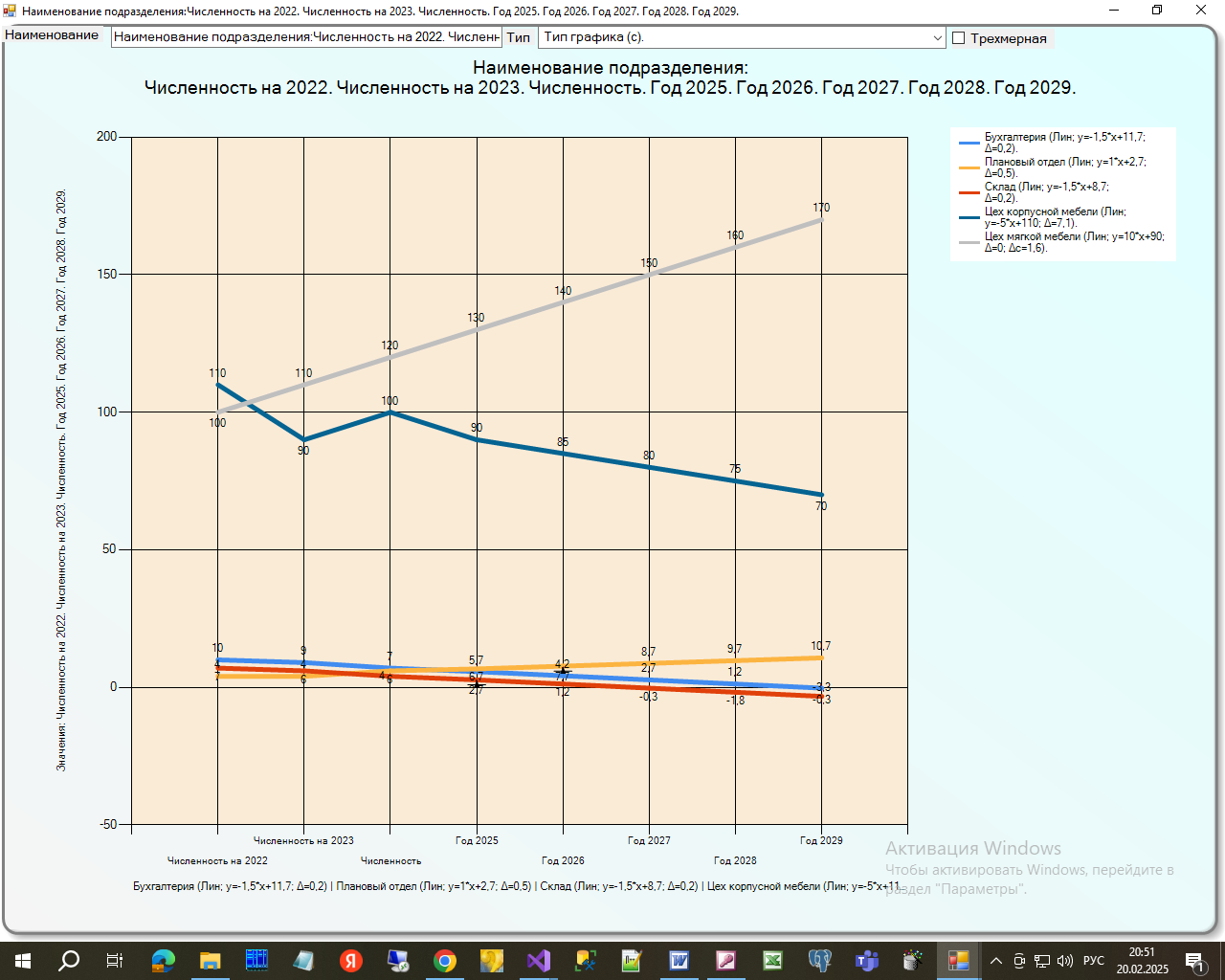
**Рисунок 7 – Конструктор таблиц с итогами.**

****

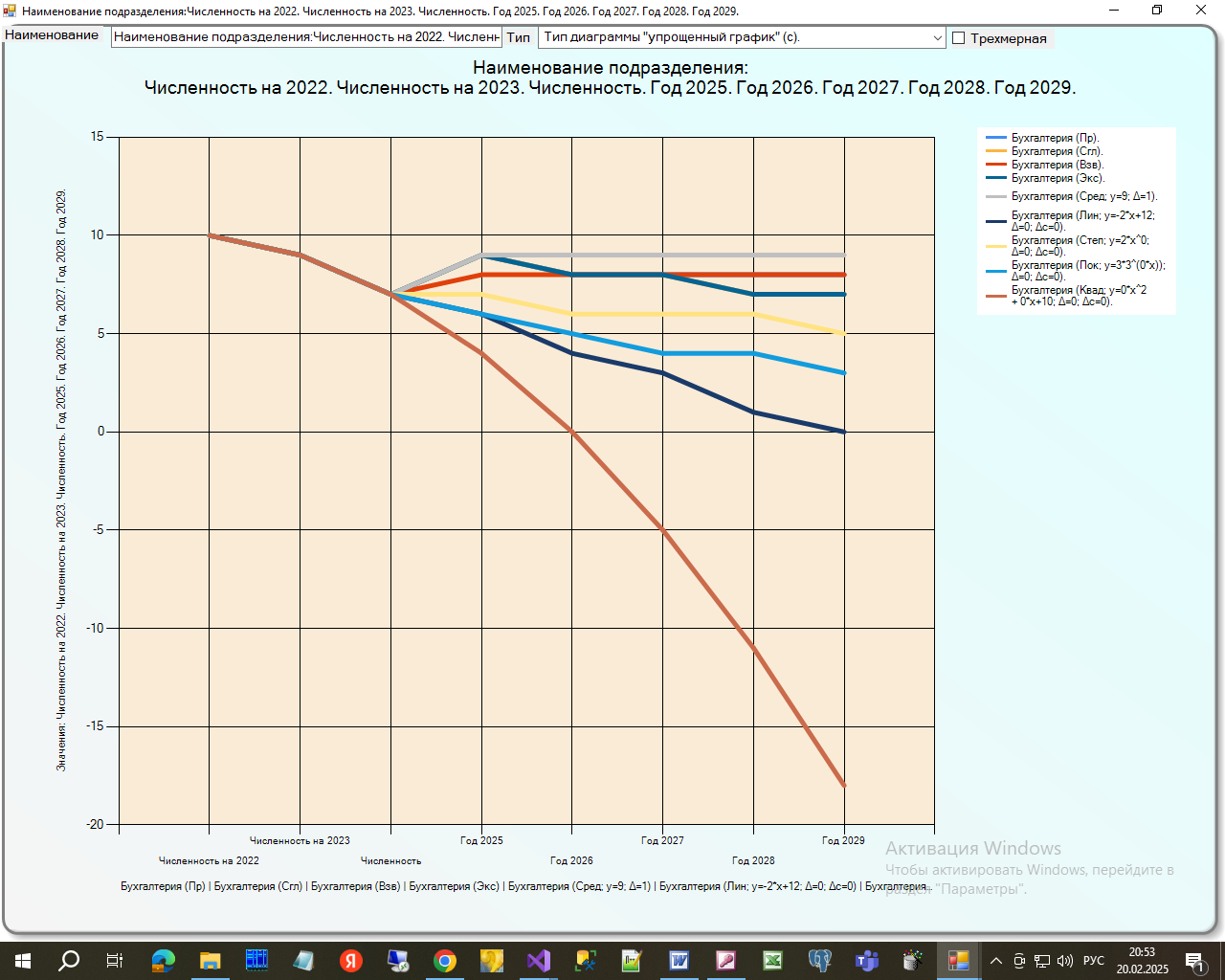
**Рисунок 8 – Конструктор таблиц с прогнозами (экстраполяцией) и аппроксимацией.**

****

**Рисунок 9 – Прогноз по одному методу в табличной форме.**

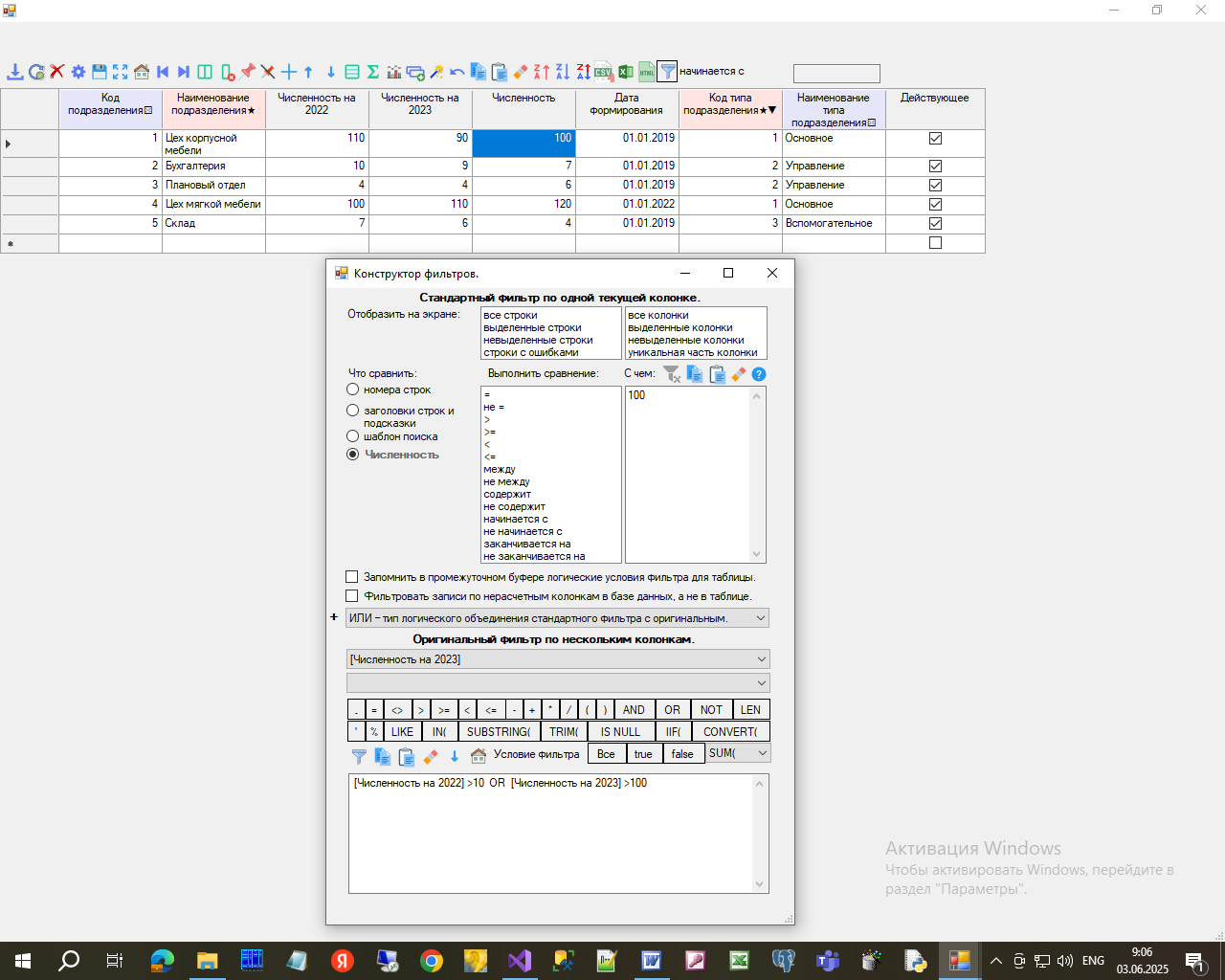
****

**Рисунок 10 – Прогноз по одному методу в графической форме.**

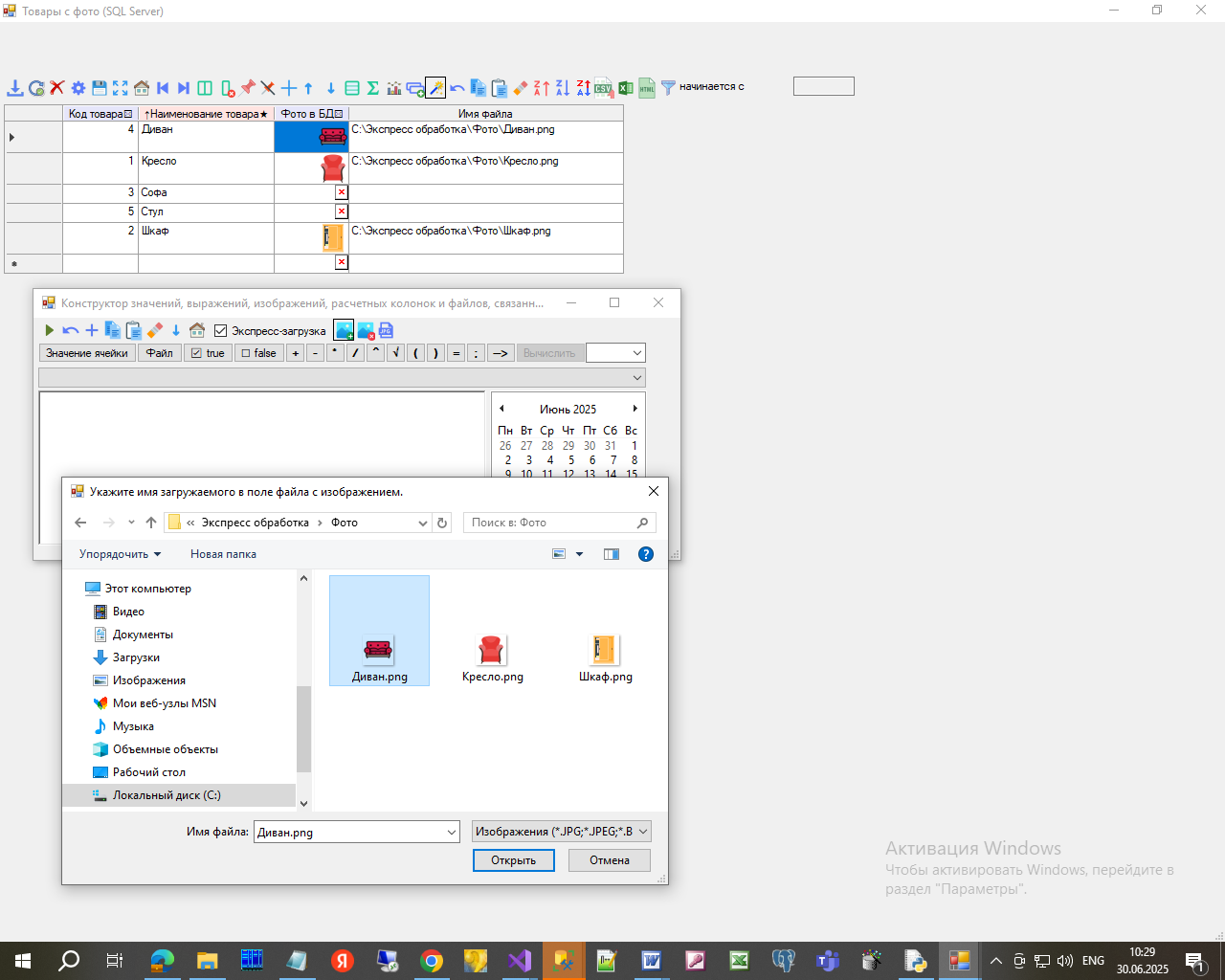
****

**Рисунок 11 – Прогноз одновременно по нескольким методам в**

**графической форме.**

****

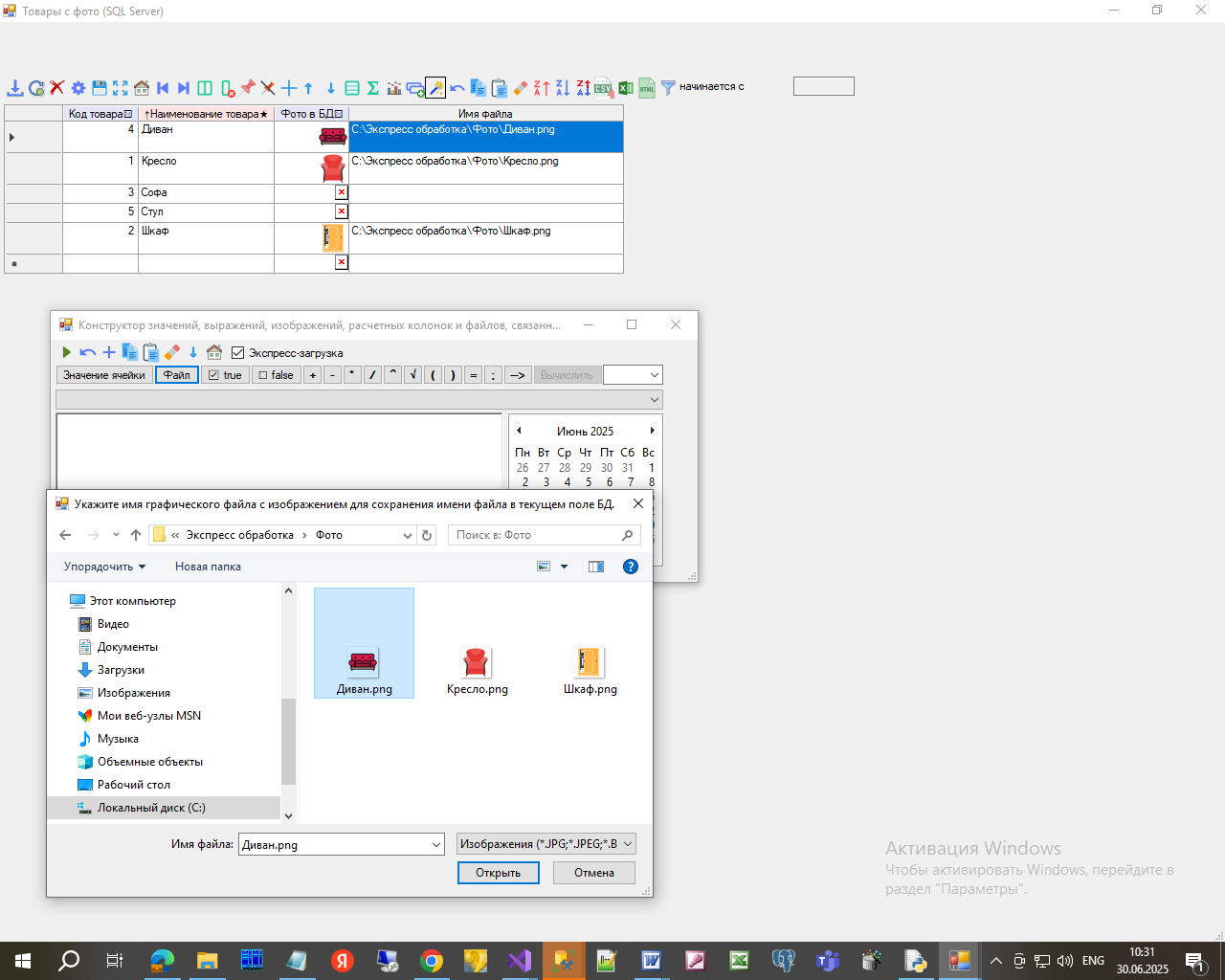
**Рисунок 12 – Конструктор фильтров.**

****

**Рисунок 13 – Конструктор значений, выражений, изображений,**

**расчетных колонок и файлов, связанных с БД.**

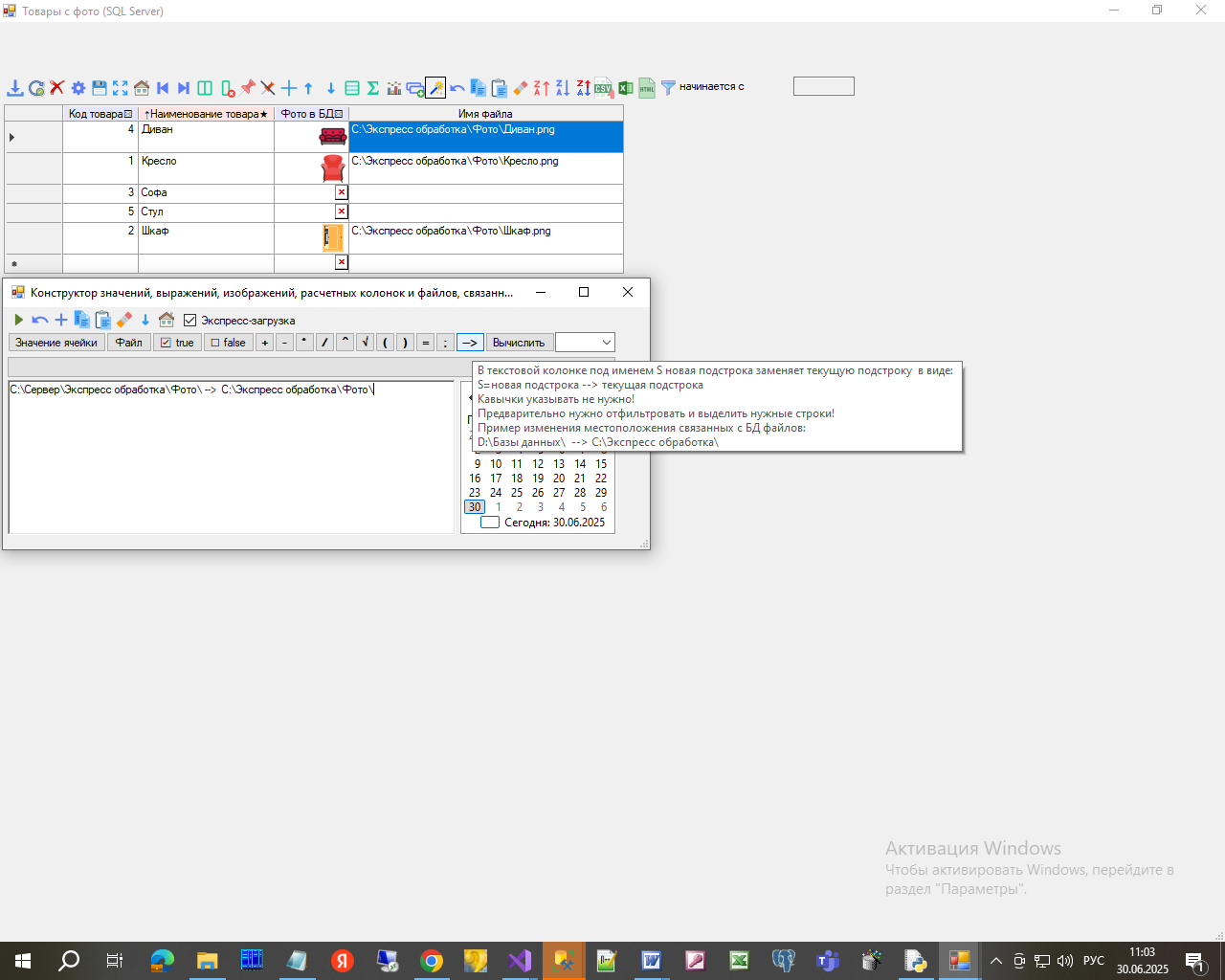
**Загрузка изображения из графического файла в поле БД.**

******

**Рисунок 14 – Конструктор значений, выражений, изображений,**

**расчетных колонок и файлов, связанных с БД.**

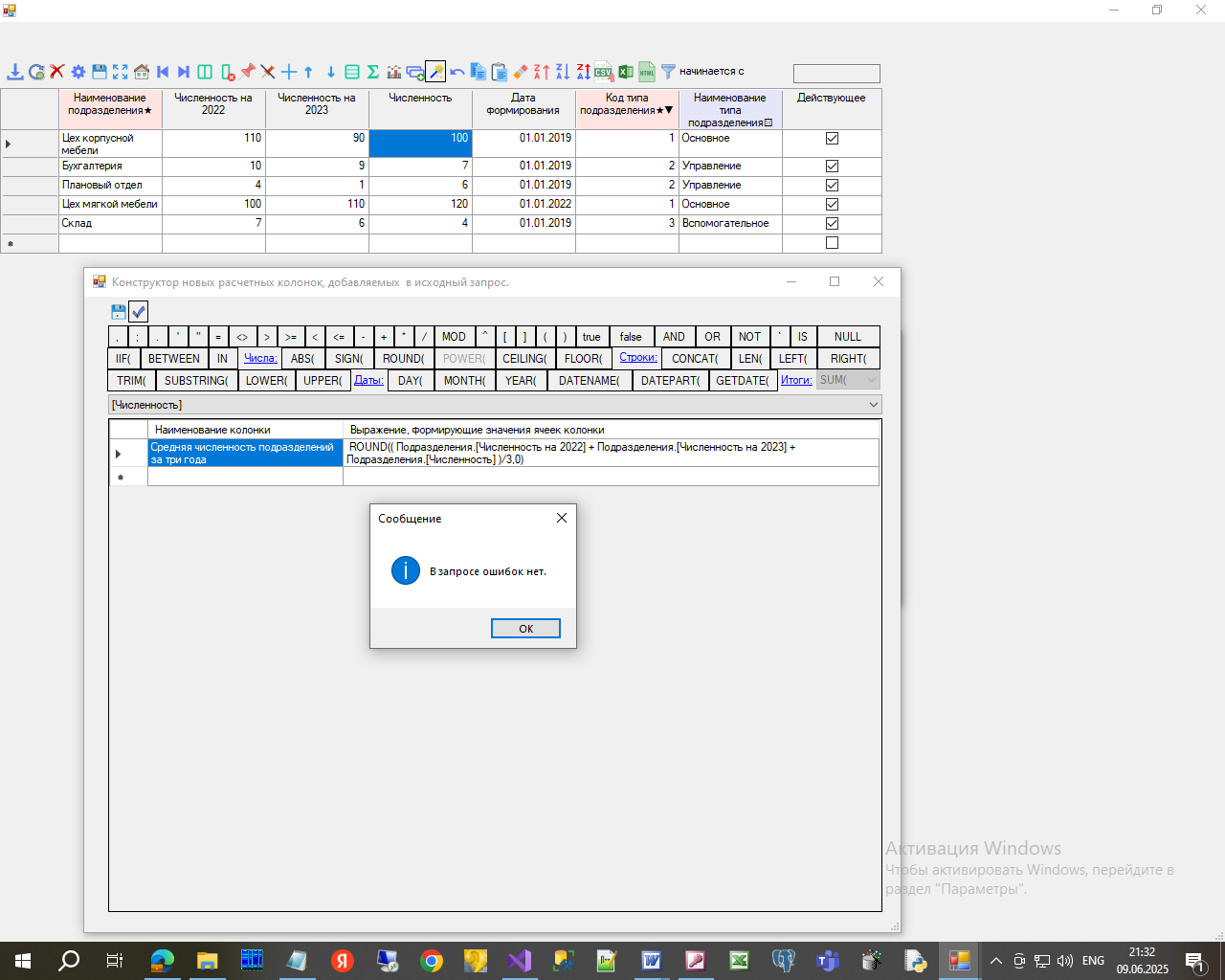
**Загрузка полного имени графического файла в БД.**

******

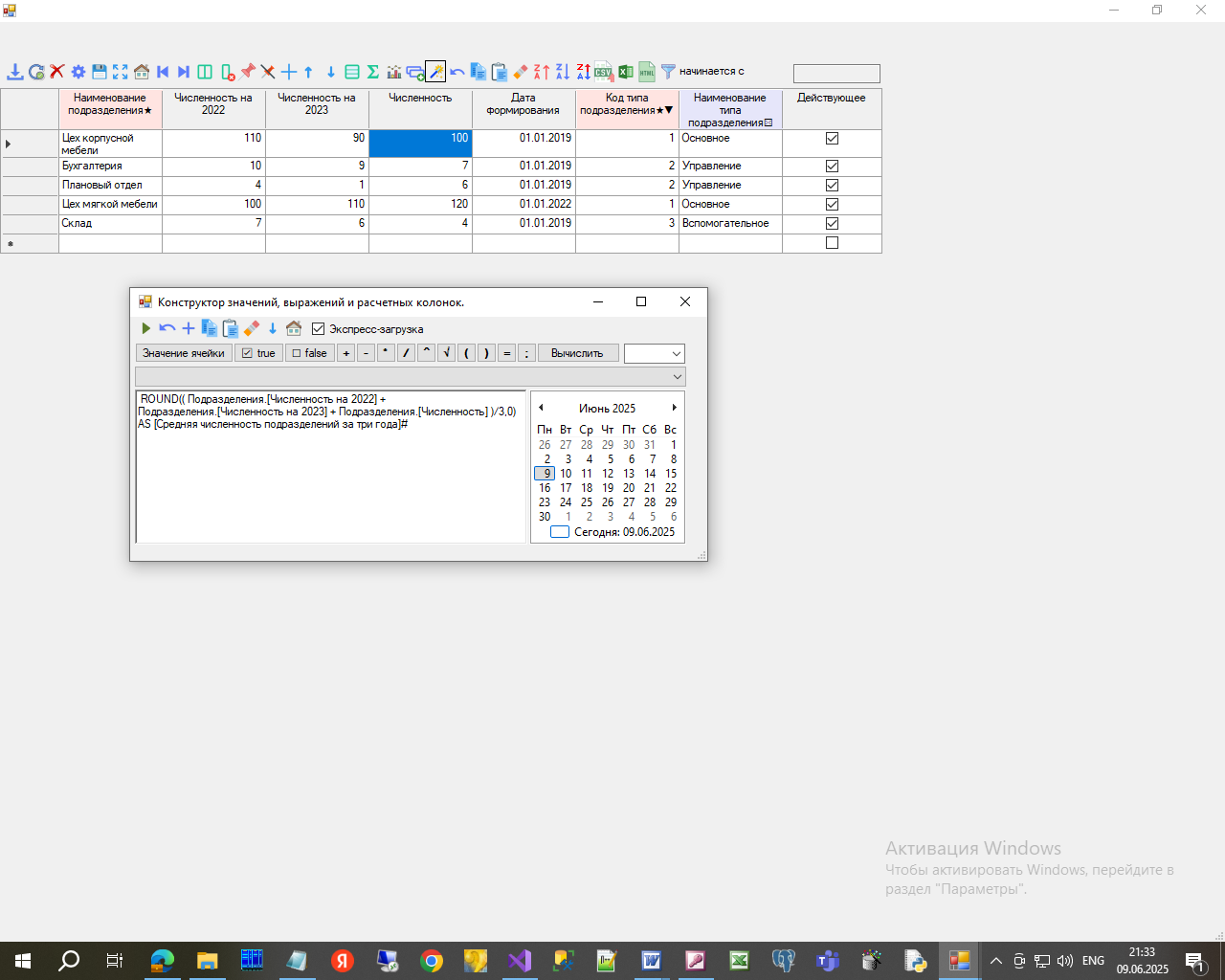
**Рисунок 15 – Конструктор значений, выражений, изображений,**

**расчетных колонок и файлов, связанных с БД.**

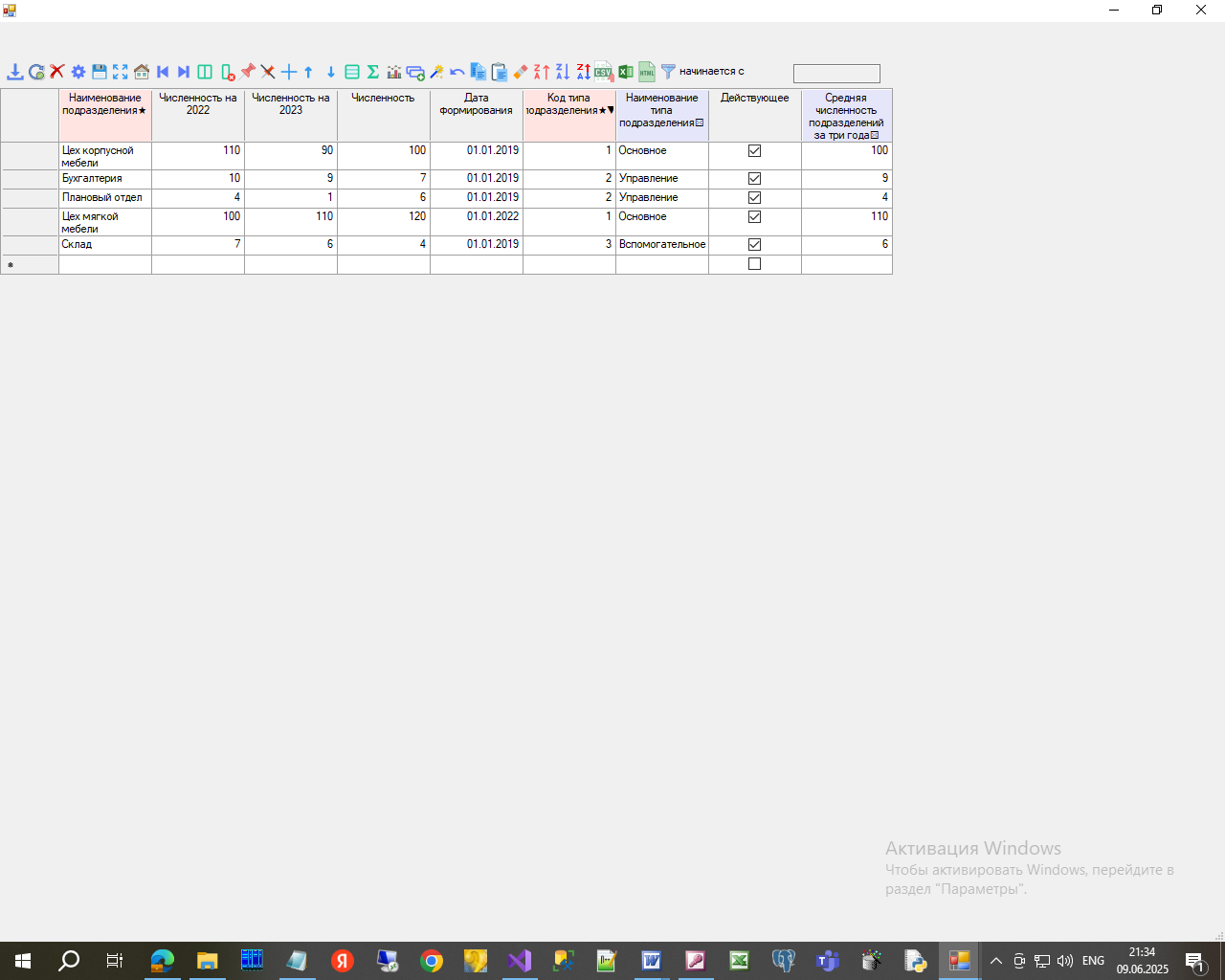
**Изменение местоположения файлов, связанных с БД.**



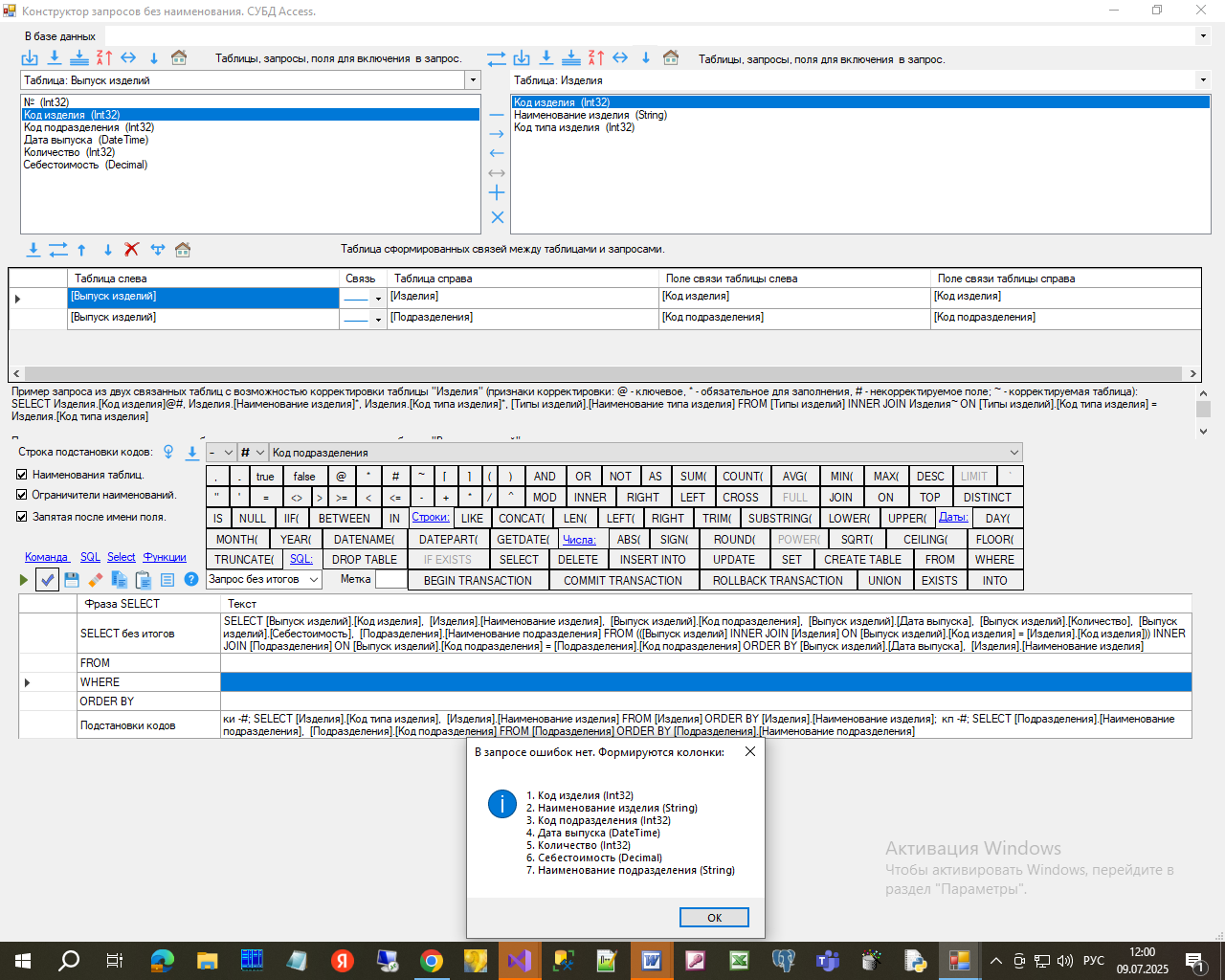
**Рисунок 16 – Конструктор формирования дополнительных расчетных колонок в исходном запросе.**



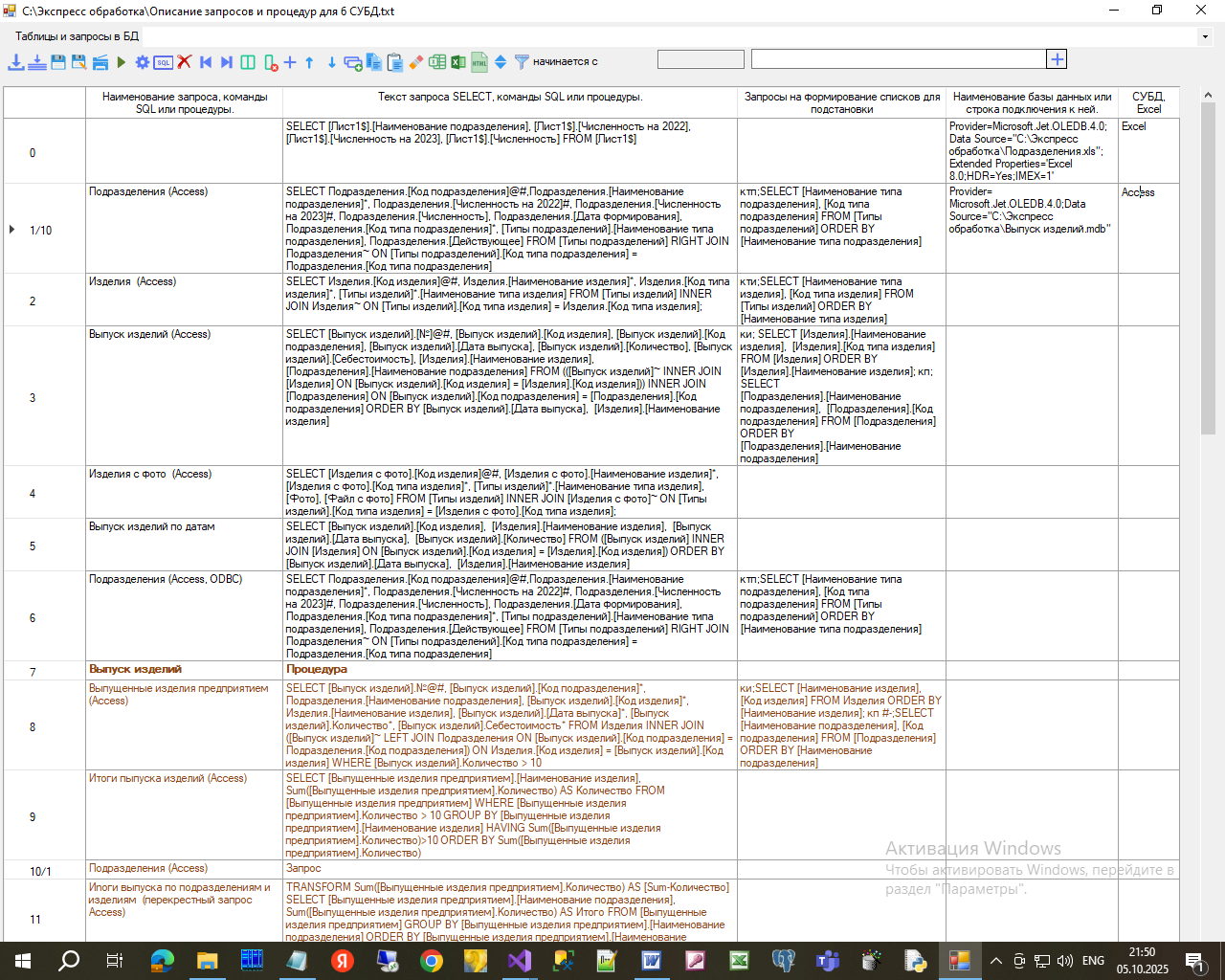
**Рисунок 16 – Описание сформированых дополнительной расчетной колонки**

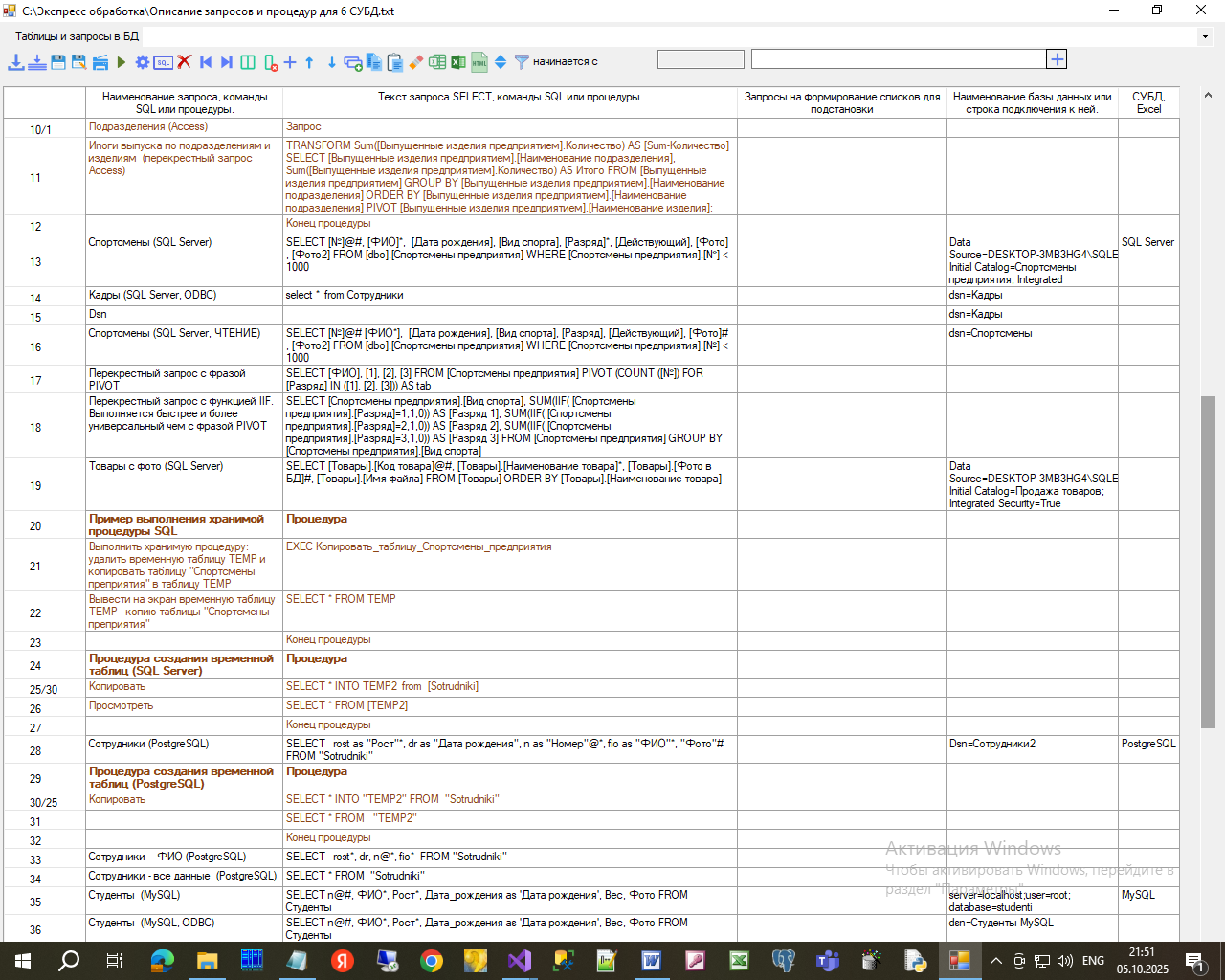


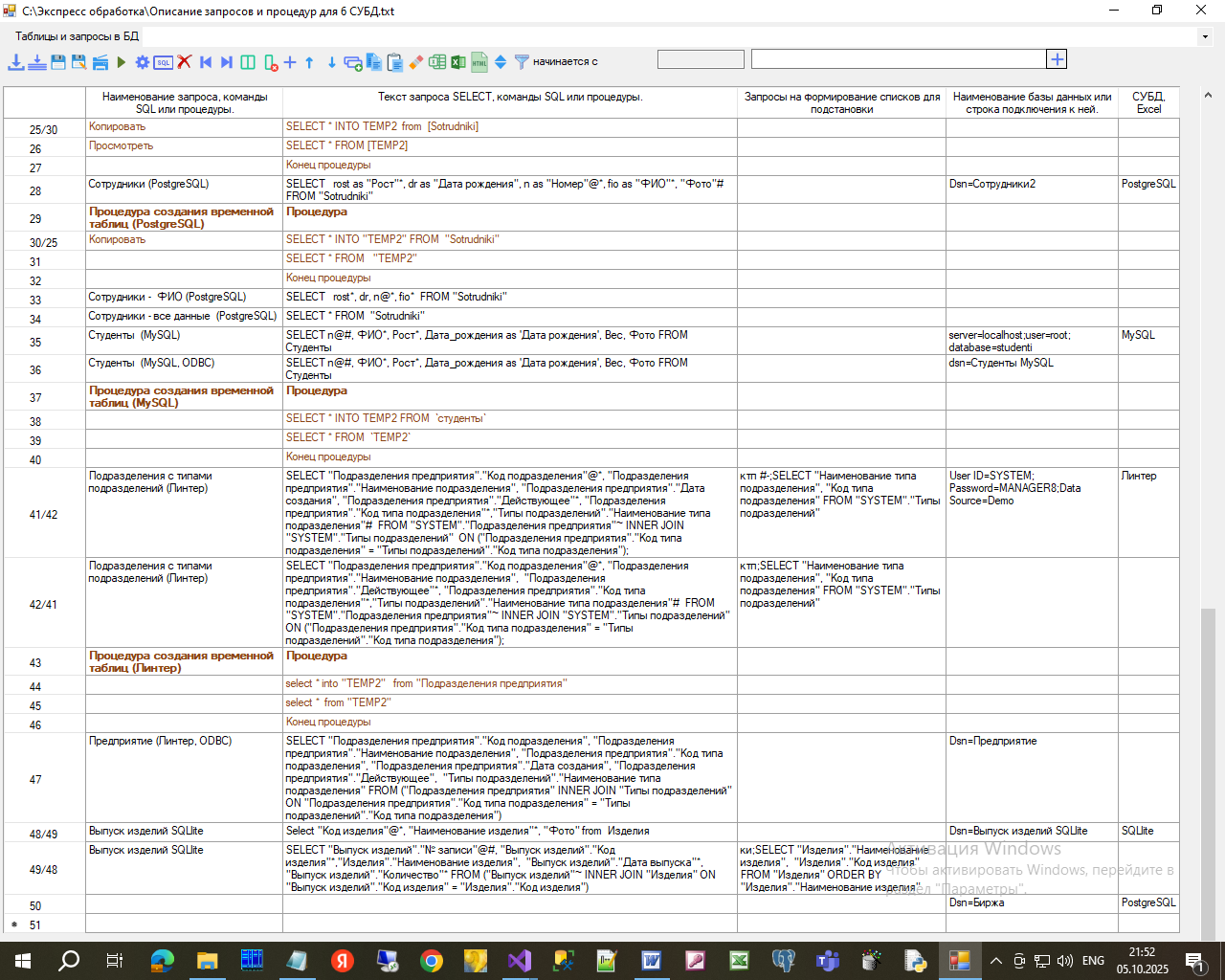
**Рисунок 18 – Сформированная таблица с расчетной колонкой**



**Рисунок 19– Конструктор запросов, процедур и команд *SQL*.**

****

****

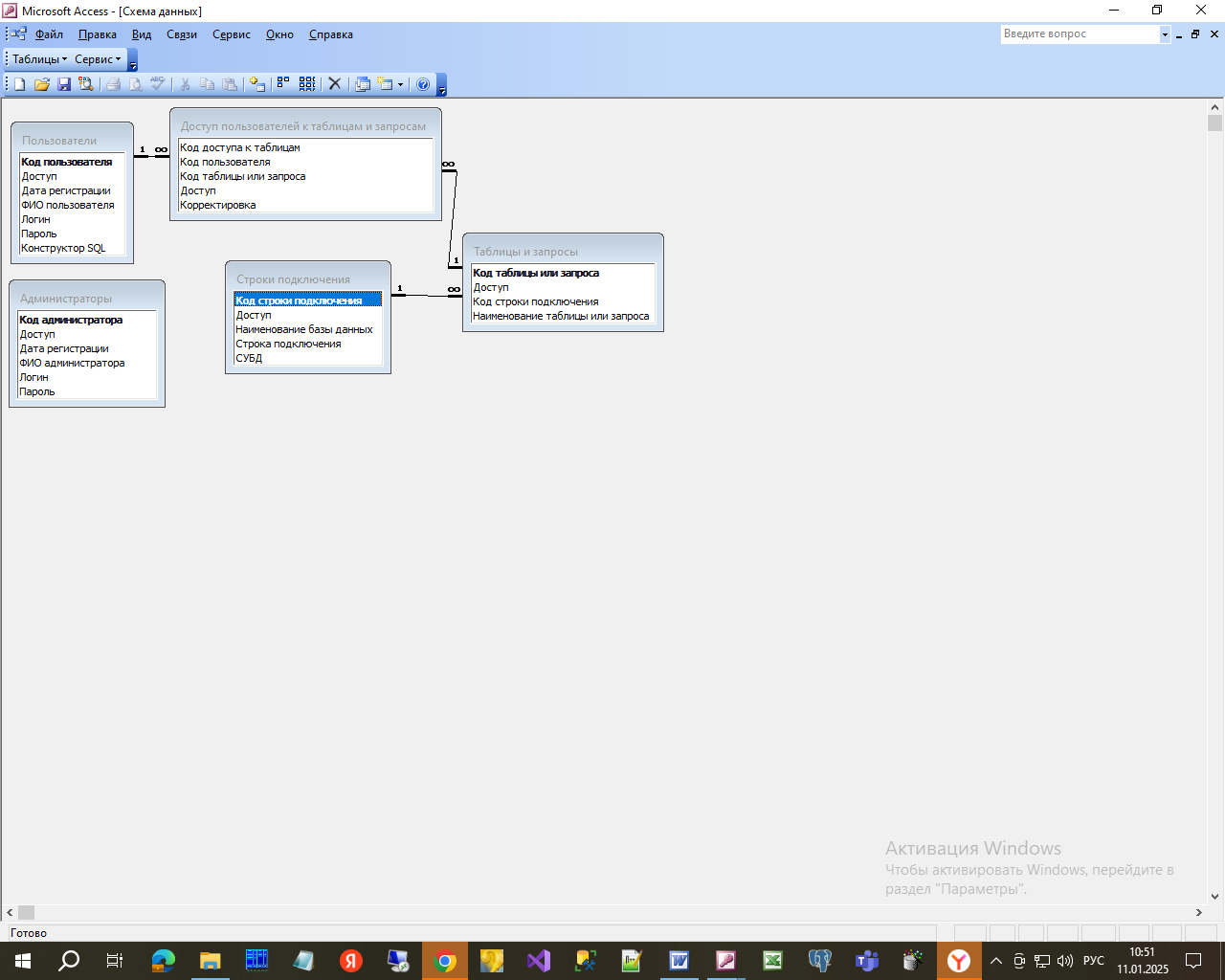
****

**Рисунок 20 – Запросы, процедуры и команды *SQL*,**

**выполняемые приложением *Экспресс\_ обработка.exe***

****

**Рисунок 21 – Окно приложения Экспресс\_обработки \_таблиц\_Excel**

****

**Рисунок 22 – Схема данных базы данных для администрирования БД**

**в приложении Экспресс\_обработка.exe.**

Доступ к файлам *Excel* не администрируется – любые файлы *Excel* доступны для пользователей.

***Примеры программ на языках С++, F#, Visual Basic, Python в которых используются средства Фреймворка DBGridChart через приложение Экспресс\_обработка\_таблицы.exe.***

Пример программы на *С++*

// Пример применения приложения Экспресс\_обработка\_таблицы.exe в программе на C++ Автор Плещев В.В. 08.01.2025

#include <string>

int main()

{

// Символ # должен быть указан перед наименованием свойства!

// Внутри строковых констант перед символами \ и " нужно указывать символ \

std::string

program = "C:\\Экспресс\_обработка\_таблицы\\bin\\Debug\\Экспресс\_обработка\_таблицы.exe", // Местоположение приложения;

p1 = "#ConnString=Provider = Microsoft.Jet.OLEDB.4.0;Data Source=\"C:\\Демонстрационный пример Экспресс обработка таблицы для C++\\Выпуск изделий.mdb\"", // Строка подключения к базе данных

p2 = "#DBMS = Access", // Тип СУБД, если не указан (p2 не указано), то определяется по строке подключения

// p3= текст команды Select с возможностью корректировки таблицы Подразделения

// Для корректировки после имен полей или их псевдонимов в корректируемой таблице указываются специальные символы-признаки:

// ключевое (@), обязательное для заполнения (\*), некорректируемое поле (#)

// и после имени корректируемой таблицы (если во фразе From несколько таблиц) символ ~

p3 = "#Select = SELECT Подразделения.[Код подразделения]@#,Подразделения.[Наименование подразделения]\*, Подразделения.[Численность на 2022], Подразделения.[Численность на 2023], Подразделения.[Численность], Подразделения.[Дата формирования], Подразделения.[Код типа подразделения]\*, [Типы подразделений].[Наименование типа подразделения], Подразделения.[Действующее] FROM [Типы подразделений] RIGHT JOIN Подразделения~ ON [Типы подразделений].[Код типа подразделения] = Подразделения.[Код типа подразделения] ORDER BY Подразделения.[Наименование подразделения]", // Запрос

// p4= текст строки подстановки кода типа подразделения в колонку под именем ктп (первые буквы имени поля с кодом) из списка, сформированного командой Select.

// После имени колонки через пробел указаны символы - режимы подстановки только из списка(-), который открывается в текущей ячейке(#)

p4 = "#Podstavit\_v\_kolonki = ктп #-; SELECT [Наименование типа подразделения], [Код типа подразделения] FROM [Типы подразделений] ORDER BY [Наименование типа подразделения]", //Подстановка кодов по наименованиям

p5 = "#Flag\_Create\_SQL\_Select = True", // Разрешить (True)/не разрешить (False) использовать конструктор команд SQL (по умолчанию - True и p5 можно не указывать)

p6 = "#Table\_№ = 2367", // Уникальный номер таблицы или запроса для сохранения макета настройки индивидуально для таблицы

parameters = p1 + p2 + p3 + p4 + p5 + p6, // параметры приложения

command = program + " " + parameters; // командная строка

system(command.c\_str()); // выполнение приложения

}

Пример программы на *F#*:

//Пример применения приложения Экспресс\_обработка\_таблицы.exe в программе на F# Автор Плещев В.В. 12.01.2025

open System.Diagnostics

[<EntryPoint>]

let main argv =

let program = """C:\Экспресс\_обработка\_таблицы\bin\Debug\Экспресс\_обработка\_таблицы.exe""" // Местоположение приложения"

let p1="""ConnString=Provider = Microsoft.Jet.OLEDB.4.0;Data Source="C:\Демонстрационный пример Экспресс обработка таблицы для F#\Выпуск изделий.mdb" """ // Строка подключения к базе данных

let p2 = "#DBMS = Access" // Тип СУБД, если не указан (p2 не указано), то опредляется по строке подключения

// p3= текст команды Select с возможностью корректировки таблицы Подразделения

// Для корректировки после имен полей или их псевдонимов к корректируемой таблице указываются специальные символы-признаки:

// ключевое (@), обязательное для заполнения (\*), некорректируемое (#) поле

// и после имени корректируемой таблицы (если во фразе From несколько таблиц) символ ~

let p3 = "#Select = SELECT Подразделения.[Код подразделения]@#,Подразделения.[Наименование подразделения]\*, Подразделения.[Численность на 2022], Подразделения.[Численность на 2023], Подразделения.[Численность], Подразделения.[Дата формирования], Подразделения.[Код типа подразделения]\*, [Типы подразделений].[Наименование типа подразделения], Подразделения.[Действующее] FROM [Типы подразделений] RIGHT JOIN Подразделения~ ON [Типы подразделений].[Код типа подразделения] = Подразделения.[Код типа подразделения] ORDER BY Подразделения.[Наименование подразделения]" // Запрос

// p4= текст строки подстановки кода типа подразделения в колонку под именем ктп (первые буквы имени поля с кодом) из списка, сформированного командой Select.

// После имени колонки через пробел указаны символы - режимы подстановки только из списка (-), который открывается в текущей ячейке (#)

let p4 = "#Podstavit\_v\_kolonki = ктп #-; SELECT [Наименование типа подразделения], [Код типа подразделения] FROM [Типы подразделений] ORDER BY [Наименование типа подразделения]" //Подстановка кодов по наименованиям

let p5 = "#Flag\_Create\_SQL\_Select = True" // Разрешить (True)/не разрешить (False) использовать конструктор команд SQL По умолчанию - True (p5 можно не указывать)

let p6 = "#Table\_№ = 2367" // Уникальный номер таблицы или запроса для сохранения макета настройки индивидуально для таблицы

let parameters = p1 + p3 + p2 + p4 + p5 + p6 // параметры приложения

let command = program + " " + parameters // командная строка

let psi = ProcessStartInfo()

psi.FileName <- program // Местоположение приложения

psi.Arguments <- parameters // строка с аргументами-свойствами

psi.Verb <- "runas" // консольная утилита runas.exe, которую можно использовать для запуска приложений от имени другого пользователя из командной строки

Process.Start(psi) // запуск приложения

0 // код возврата

Пример программы на *Visual Basic*:

' Пример применения приложения Экспресс\_обработка\_таблицы.exe в программе на Visual Basic Автор Плещев В.В. 08.01.2025

Public Class Form1

Private Sub ButtonVipolnit\_Click(sender As Object, e As EventArgs) Handles ButtonVipolnit.Click

' Символ # должен быть указан перед наименованием свойства!

' Внутри строковых констант вместо одной кавычки указываются две кавычки!

Dim program = "C:\\Экспресс\_обработка\_таблицы\\bin\\Debug\\Экспресс\_обработка\_таблицы.exe" ' Местоположение приложения

Dim p1 = "#ConnString=Provider = Microsoft.Jet.OLEDB.4.0;Data Source=""C:\Демонстрационный пример Экспресс обработка таблицы для Visual Basic\Выпуск изделий.mdb""" ' Строка подключения к базе данных

Dim p2 = "#DBMS = Access" ' Тип СУБД, если не указан (p2 не указано), то определяется по строке подключения

' p3= текст команды Select с возможностью корректировки таблицы Подразделения

' Для корректировки после имен полей или их псевдонимов к корректируемой таблице указываются специальные символы-признаки:

' ключевое (@), обязательное для заполнения (\*), некорректируемое поле (#)

' и после имени корректируемой таблицы (если во фразе From несколько таблиц) символ ~

Dim p3 = "#Select = SELECT Подразделения.[Код подразделения]@#,Подразделения.[Наименование подразделения]\*, Подразделения.[Численность на 2022], Подразделения.[Численность на 2023], Подразделения.[Численность], Подразделения.[Дата формирования], Подразделения.[Код типа подразделения]\*, [Типы подразделений].[Наименование типа подразделения], Подразделения.[Действующее] FROM [Типы подразделений] RIGHT JOIN Подразделения~ ON [Типы подразделений].[Код типа подразделения] = Подразделения.[Код типа подразделения] ORDER BY Подразделения.[Наименование подразделения]" ' Запрос

' p4= текст строки подстановки кода типа подразделения в колонку под именем ктп (первые буквы имени поля с кодом) из списка, сформированного командой Select.

' После имени колонки через пробел указаны символы - режимы подстановки только из списка (-), который открывается в текущей ячейке (#)

Dim p4 = "#Podstavit\_v\_kolonki = ктп #-; SELECT [Наименование типа подразделения], [Код типа подразделения] FROM [Типы подразделений] ORDER BY [Наименование типа подразделения]" ' Подстановка кодов по наименованиям

Dim p5 = "#Flag\_Create\_SQL\_Select = True" ' Разрешить(True) / не разрешить (False) использовать конструктор команд SQL (по умолчанию - True и p5 можно не указывать)

Dim p6 = "#Table\_№ = 2367" ' Уникальный номер таблицы или запроса для сохранения макета настройки индивидуально для таблицы

Dim parameters = p1 + p2 + p3 + p4 + p5 + p6 ' параметры приложения

Dim psi As New ProcessStartInfo ' создание объекта для запуска приложения

psi.FileName = program ' Местоположение приложения

psi.Arguments = parameters ' строка с аргументами-свойствами

psi.Verb = "runas" ' консольная утилита runas.exe, которую можно использовать для запуска приложений от имени другого пользователя из командной строки

Process.Start(psi) ' запуск приложения

End Sub

End Class

Пример программы на *Python:*

# Пример применения приложения Экспресс\_обработка\_таблицы.exe

# в программе на Python Автор Плещев В.В. 13.01.2025

# Символ # должен быть указан перед наименованием свойства!

# Внутри строковых констант вместо одной кавычки указываются две кавычки!

**import** os # подключение библиотеки OS

program = "C:\\Экспресс\_обработка\_таблицы\\bin\\Debug\\Экспресс\_обработка\_таблицы.exe" # Местоположение выполняемого приложения Экспресс\_обработка\_таблицы.exe

p1 = "#ConnString=Provider = Microsoft.Jet.OLEDB.4.0;Data Source=""C:\Демонстрационный пример Экспресс обработка таблицы для Python\Выпуск изделий.mdb""" # Строка подключения к базе данных

p2 = "#DBMS = Access" # Тип СУБД, если не указан (p2 не указано), то определяется по строке подключения

# p3= текст команды Select с возможностью корректировки таблицы Подразделения

# Для корректировки после имен полей или их псевдонимов к корректируемой таблице указываются специальные символы-признаки:

# ключевое (@), обязательное для заполнения (\*), некорректируемое (#) поле

# и после имени корректируемой таблицы (если во фразе From несколько таблиц) символ ~

p3 = "#Select = SELECT Подразделения.[Код подразделения]@#,Подразделения.[Наименование подразделения]\*, Подразделения.[Численность на 2022], Подразделения.[Численность на 2023], Подразделения.[Численность], Подразделения.[Дата формирования], Подразделения.[Код типа подразделения]\*, [Типы подразделений].[Наименование типа подразделения], Подразделения.[Действующее] FROM [Типы подразделений] RIGHT JOIN Подразделения~ ON [Типы подразделений].[Код типа подразделения] = Подразделения.[Код типа подразделения] ORDER BY Подразделения.[Наименование подразделения]" # Запрос

# p4= текст строки подстановки кода типа подразделения в колонку под именем ктп (первые буквы имени поля с кодом) из списка, сформированного командой Select.

# После имени колонки через пробел указаны символы - режимы подстановки только из списка (-), который открывается в текущей ячейке (#)

p4 = "#Podstavit\_v\_kolonki = ктп #-; SELECT [Наименование типа подразделения], [Код типа подразделения] FROM [Типы подразделений] ORDER BY [Наименование типа подразделения]" # Подстановка кодов по наименованиям

p5 = "#Flag\_Create\_SQL\_Select = True" # Разрешить(True) / не разрешить (False) использовать конструктор команд SQL (по умолчанию - True и p5 можно не указывать)

p6 = "#Table\_№ = 2367" # Уникальный номер таблицы или запроса для сохранения макета настройки индивидуально для таблицы

parameters = p1 + p2 + p3 + p4 + p5 + p6 # параметры приложения

os.system(program + " " + parameters)

**input**()

Пример программы на *Java*

**import** javax.swing.\*;  *//библиотека для GUI*

**import** java.awt.\*;   *//базовые классы для графики и менеджеры компоновки*

**import** java.awt.event.ActionEvent; *//обработка функций кнопок*

**import** java.io.IOException;      *//для обработки ошибок ввода и вывода*

**public** **class** Main {

**public** **static** **void** main(**String**[] args) {

*//Местоположение выполняемого приложения Экспресс\_обработка\_таблицы.exe*

**String** program = "C:**\\**Экспресс\_обработка\_таблицы**\\**bin**\\**Debug**\\**Экспресс\_обработка\_таблицы.exe";

*//Строка подключения к базе данных*

**String** p1 = "#ConnString=Provider = Microsoft.Jet.OLEDB.4.0;Data Source=**\"**C:**\\**Демонстрационный пример Экспресс обработка таблицы для Java**\\**Выпуск изделий.mdb**\"**";

**String** p2 = "#DBMS = Access"; *//Тип СУБД, если не указан (p2 не указано), то определяется по строке подключения*

*//p3= текст команды Select*

*//Для корректировки после имен полей или их псевдонимов к корректируемой таблице указываются специальные символы-признаки:*

*//ключевое (@), обязательное для заполнения (\*), некорректируемое (#) поле*

*//и после имени корректируемой таблицы (если во фразе From несколько таблиц) символ ~*

**String** p3 = "#Select = SELECT Подразделения.[Код подразделения]@#,Подразделения.[Наименование подразделения]\*, Подразделения.[Численность на 2022], Подразделения.[Численность на 2023], Подразделения.[Численность], Подразделения.[Дата формирования], Подразделения.[Код типа подразделения]\*, [Типы подразделений].[Наименование типа подразделения], Подразделения.[Действующее] FROM [Типы подразделений] RIGHT JOIN Подразделения~ ON [Типы подразделений].[Код типа подразделения] = Подразделения.[Код типа подразделения] ORDER BY Подразделения.[Наименование подразделения]";

*//p4= текст строки подстановки кода типа подразделения в колонку под именем ктп (первые буквы имени поля с кодом) из списка, сформированного Select.*

*//После имени колонки через пробел указаны символы - режимы подстановки только из списка (-), который открывается в текущей ячейке (#)*

**String** p4 = "#Podstavit\_v\_kolonki = ктп #-; SELECT [Наименование типа подразделения], [Код типа подразделения] FROM [Типы подразделений] ORDER BY [Наименование типа подразделения]";

**String** p5 = "#Flag\_Create\_SQL\_Select = True"; *//Разрешить(True) / не разрешить (False) использовать конструктор команд SQL (по умолчанию - True и p5 можно не указывать)*

**String** p6 = "#Table\_№ = 2367"; *//Уникальный номер таблицы или запроса для сохранения макета настройки индивидуально для таблицы*

*//собираем параметры в одну строку*

**String** parameters = p1 + " " + p2 + " " + p3 + " " + p4 + " " + p5 + " " + p6;

*//создание окна*

**JFrame** frame = **new** **JFrame**("Экспресс обработка таблицы");

        frame.setSize(250, 150);

        frame.setDefaultCloseOperation(**JFrame**.EXIT\_ON\_CLOSE);

        frame.setLayout(**new** **FlowLayout**()); *//задает расположение элементов в окнесоздание кнопки и ее размеров*

**JButton** button = **new** **JButton**("ВЫПОЛНИТЬ");

        button.setPreferredSize(**new** **Dimension**(200, 40));

*//обработка нажатия на кнопку*

        button.addActionListener((**ActionEvent** e) -> {

            frame.dispose(); *//закрыть текущее окно*

**try** {

*//объект для запуска внешней программы*

**ProcessBuilder** pb = **new** **ProcessBuilder**(program, parameters);

                pb.inheritIO(); *//наследуем стандартные потоки вывода (чтобы видеть вывод exe в консоли Java)*

                pb.start(); *//асинхронный запуск процесса*

            }

**catch** (**IOException** ex) {

                ex.printStackTrace(); *//в случае ошибки (допустим, .exe не найден) выводим сообщение об ошибке*

**JOptionPane**.showMessageDialog(**null**, "Ошибка запуска программы!", "Ошибка!", **JOptionPane**.ERROR\_MESSAGE);

            }

            });

*//добавляем кнопку в окно, центрируем его на экране и делаем окно видимым*

        frame.add(button);

        frame.setLocationRelativeTo(**null**);

        frame.setVisible(**true**);

        }

    }

}

Аналогичные программы можно разработать и для других языков программирования.