Лабораторная работа 3

ОРГАНИЗАЦИЯ СВЯЗИ MASTER-DETAIL МЕЖДУ ДАННЫМИ

Цель работы: разработать клиентское приложение баз данных, позволяющее работать со связанными данными по принципу Master–Detail

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Ознакомиться с теоретическими сведениями.
2. Открыть копию проекта лабораторной работы №2.
3. Создать необходимые представления.
4. Добавить к проекту компоненты для работы с новыми представлениями.
5. Настроить между данными связь **Master**–**Detail**.
6. Организовать выбор типа операции при помощи **RadioGroup**.
7. Реализовать редактирование списка товаров в определенной операции.
8. Ответить на контрольные вопросы.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Master–Detail представление связанных данных при котором основные данные отображаются в виде списка, а их подробное описание или связанные данные только при выборе конкретного значения из этого списка. Например, так обычно отображается информация в почтовых клиентах.

3. РАБОТА СО СВЯЗАННЫМИ ТАБЛИЦАМИ ПРИ ПОМОЩИ MASTER–DETAIL

Для реализации текущего задания необходимо сделать копию проекта, сформированного в предыдущей лабораторной работе. В текущей версии программы отображается представление об операциях с товарами (рис. 3.1). Однако приложение не показывает на какую сумму выполнены операции и не ясно какие товары участвовали в операциях.

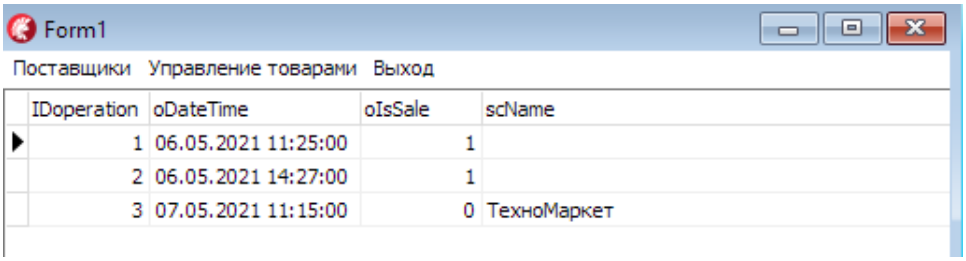

Рис. 3.1. Конструирование запроса для нового представления

Сформируем в БД новое представление **view_operation_goods_list**, которое покажет, какие товары (с учетом типа) и в каком количестве участвовали в операциях. При этом, также будем отображать цену товара.

Для этого необходимо сгенерировать *sql*-запрос. Будем использовать конструктор запросов. Перенесите в область конструктора таблицы **op_goods_list**, **goods_catalog** и **type_goods**. Выберите поля как показано на рис. 3.2.

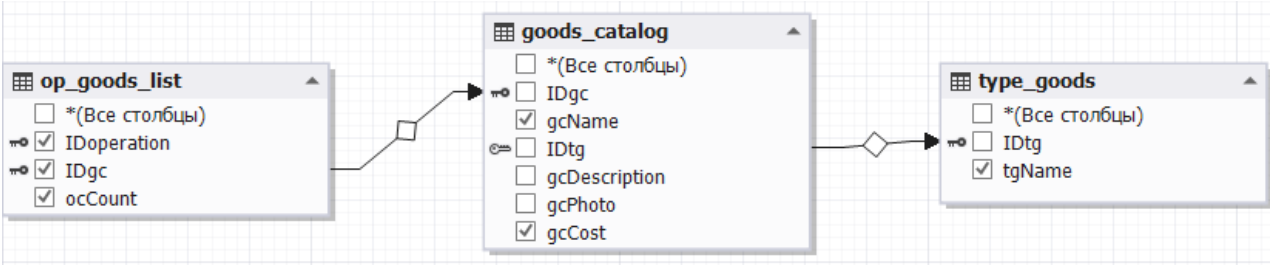

Рис. 3.2. Конструирование запроса для нового представления

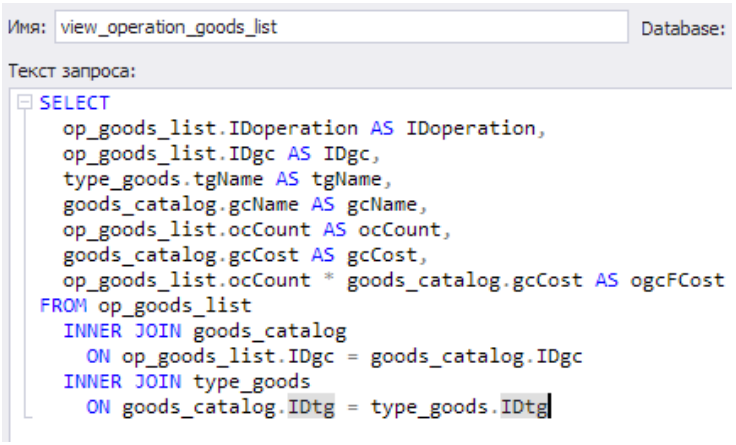

Рис. 3.3. *sql*-запрос нового представления

IDoperation INT(11)	IDgc INT(11)	tgName VARCHAR(50)	gcName VARCHAR(100)	ocCount INT(11)	gcCost DECIMAL(10, 2)
1	1	Ноутбуки	Acer Aspire 1 A115-32-C2Z1 серебристый	1	25499,00
3	2	Ноутбуки	Lenovo IdeaPad 3 15IGL05 серый	5	27955,00
3	3	Ноутбуки	ASUS VivoBook M513UA-BQ080 черный	2	49999,00
3	4	Системные блоки	DEXP Mars E317	4	41999,00
2	5	Системные блоки	ASUS ROG Strix GL10CS-RU077T	1	64990,00
2	7	Мониторы	HP V28 4K	2	19399,00

Рис. 3.4. Выводимые данные представлением *view_operation_goods_list*

Теперь необходимо отобразить полную стоимость товаров, участвовавших в конкретной операции. Для этого необходимо сделать группировку данных с помощью параметра **GROUP BY** оператора **SELECT**. Параметр **GROUP BY** используется для определения групп выходных строк, к которым могут применяться агрегатные функции (**COUNT**, **MIN**, **MAX**, **AVG**, **SUM** и др.). Если при наличии параметра **GROUP BY**, в операторе **SELECT** отсутствуют агрегатные функции, то запрос просто вернет по одной строке из каждой группы.

Поскольку представление **view_operation_goods_list** содержит все необходимые данные то мы будем использовать его для построения запроса отображающего полную стоимость товаров, участвующих в операции.

SQL код запроса будет выглядеть следующим образом, а результат выполнения представлен на рис. 3.5:

```
SELECT
  IDoperation,
  SUM(ocCount * gcCost) AS oFullCost
FROM view_operation_goods_list
GROUP BY IDoperation
```

	IDoperation INT(11)	FullCost DECIMAL(42, 2)
▶	1	25499,00
	2	103788,00
	3	407769,00

Рис. 3.5. Общая стоимость товаров по определенным операциям.

Сохраним код данного запроса как представление с именем **view_operations_fullcost**.

Теперь изменим представление **view_goods_operations**. Для этого войдем в режим дизайнера и добавим туда созданное представление (рис. 3.6). При помощи зажатой левой кнопки мыши соединим поля **IDoperation** из таблицы **operations** с представлением **view_operations_fullcost,** а также отметим поле **oFullCost** для вывода. В настройке связи необходимо указать **Выбрать все строки из Operations**.

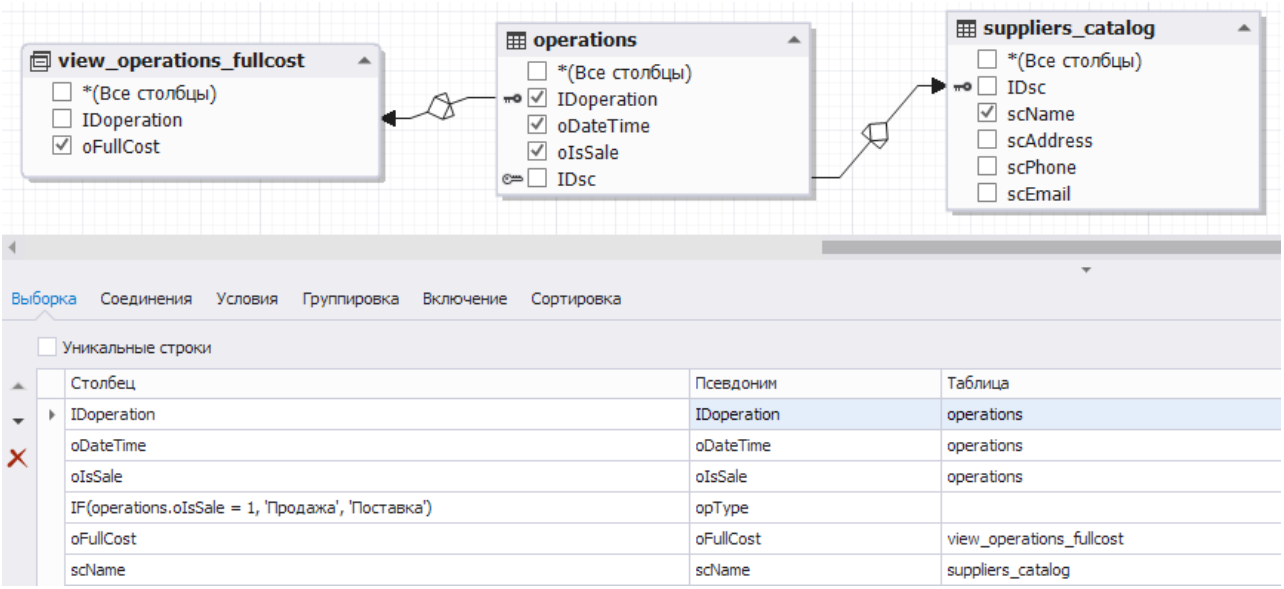

Рис. 3.6. Модификация представления *view_goods_operations*

Обратите внимание, что при выводе столбца **oIsSale** используется функция **IF**:

```
IF(условие, значение1, значение2)
IF(operations.oIsSale = 1, 'Продажа', 'Поставка') AS opType
```

MySQL (как и другие СУБД) предоставляют большое количество встроенных функций для работы с данными из языка *SQL*.

Имя: view_goods_operations Database: goodsmanager

Текст запроса:

```
SELECT
  operations.IDoperation AS IDoperation,
  operations.oDateTime AS oDateTime,
  operations.oIsSale AS oIsSale,
  IF(operations.oIsSale = 1, 'Продажа', 'Поставка') AS opType,
  view_operations_fullcost.oFullCost AS oFullCost,
  suppliers_catalog.scName AS scName
FROM operations
  LEFT OUTER JOIN view_operations_fullcost
    ON operations.IDoperation = view_operations_fullcost.IDoperation
  LEFT OUTER JOIN suppliers_catalog
    ON operations.IDsc = suppliers_catalog.IDsc
```

Рис. 3.7. Текст запроса представления *view_goods_operations*

Теперь в качестве типа операции будет отображаться текстовая запись и общая стоимость товаров в этой операции.

goodsmanager.view_goods_operations

Текст запроса | Дизайнер запроса | **Данные** | SQL

1000

IDoperation INT(11)	oDateTime DATETIME	oIsSale TINYINT(1)	opType VARCHAR(8)	oFullCost DECIMAL(42, 2)	scName VARCHAR(50)
1	06.05.2021 11:23:00	1	Продажа	549674,00	*(null)*
2	05.05.2021 2:22:00	1	Продажа	129287,00	*(null)*
3	07.05.2021 11:15:00	0	Поставка	397287,00	ТехноМаркет

Рис. 3.8. Данные *view_goods_operations*

Создадим еще одно представление, которое в будущем позволит компактно отображать информацию о товаре и его цене, например, в выпадающих списках.

Создайте новое представление **v_selector_goods** со следующим кодом:

```
SELECT
  IDgc,
  CONCAT(gcName, ' [', gcCost, ' руб.]') AS sNameCost
FROM goods_catalog
```

Результат этого запроса показан на рис. 3.9. Применяемая в запросе функция **CONCAT** предназначена для сложения строк при выборке из базы данных. Строками обычно выступают поля таблиц, но могут быть использованы и строковые константы.

IDgc INT(11)	sNameCost VARCHAR(120)
1	Acer Aspire 1 A115-32-C2Z1 серебристый [25499.00 руб.]
2	Lenovo IdeaPad 3 15IGL05 серый [27955.00 руб.]
3	ASUS VivoBook M513UA-BQ080 черный [49999.00 руб.]
4	DEXP Mars E317 [41999.00 руб.]
5	ASUS ROG Strix GL10CS-RU077T [64990.00 руб.]
6	Samsung F27T450FQI [15199.00 руб.]
7	HP V28 4K [19399.00 руб.]

Рис. 3.9. Данные *v_selector_goods*

В будущем для контроля значений в разрабатываемом приложении нам понадобятся сведения о количестве товара доступных на складе. Чтобы получить эти данные добавим в представление **v_selector_goods** столбец **sCount** из таблицы **storage** (рис. 3.10):

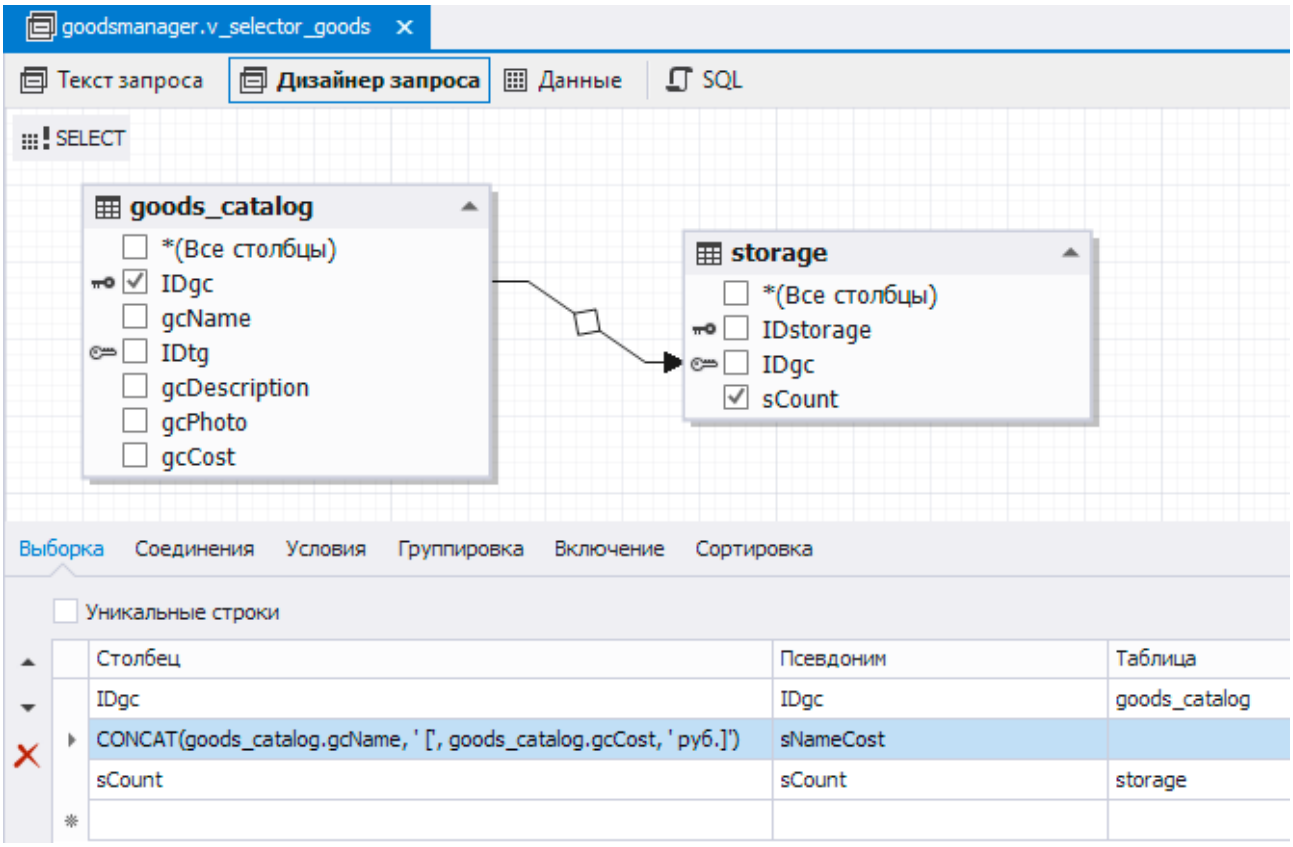

Рис. 3.10. Добавление к *v_selector_goods* столбца *sCount*

После применения изменений будут выводиться следующие данные, представленные на рис. 3.11:

IDgc INT(11)	sNameCost VARCHAR(120)	sCount INT(11)
1	Acer Aspire 1 A115-32-C2Z1 серебристый [25499.00 руб.]	4
2	Lenovo IdeaPad 3 15IGL05 серый [27955.00 руб.]	7
3	ASUS VivoBook M513UA-BQ080 черный [49999.00 руб.]	6
4	DEXP Mars E317 [41999.00 руб.]	4
5	ASUS ROG Strix GL10CS-RU077T [64990.00 руб.]	5
6	Samsung F27T450FQI [15199.00 руб.]	6
7	HP V28 4K [19399.00 руб.]	7

Рис. 3.11. Данные *v_selector_goods*

Осталось создать последнее представление **view_storage**, позволяющее просматривать сведения о товарах находящихся на складе (рис. 3.12):

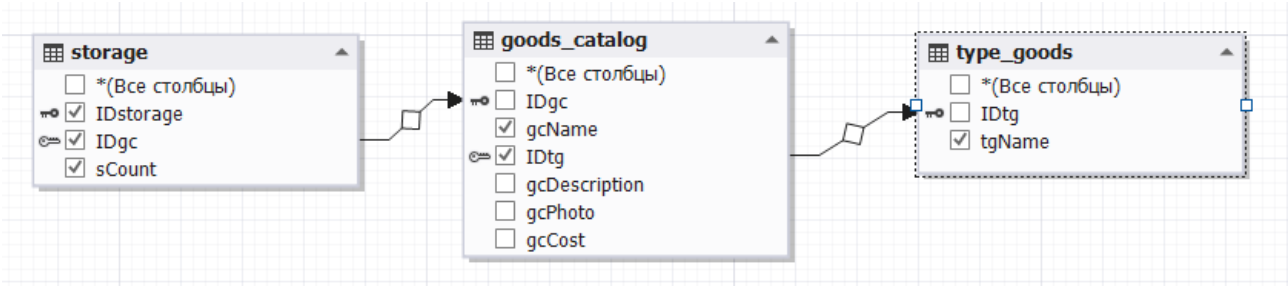

Рис. 3.12. Дизайн представления *view_storage*

Код представления будет выглядеть следующим образом:

```
SELECT
  storage.IDstorage AS IDstorage,
  storage.IDgc AS IDgc,
  goods_catalog.gcName AS gcName,
  goods_catalog.IDtg AS IDtg,
  type_goods.tgName AS tgName,
  storage.sCount AS sCount
FROM storage
  INNER JOIN goods_catalog
    ON storage.IDgc = goods_catalog.IDgc
  INNER JOIN type_goods
    ON goods_catalog.IDtg = type_goods.IDtg
```

3.1 Добавление наборов данных представлений

Для вновь созданных представлений в нашем программном проекте на **DataModule2** необходимо вынести несколько компонентов **ADOQuery** и **DataSource**. Настройка будет происходить аналогично тому, как это было сделано во второй лабораторной работе.

В связи с тем, что были внесены изменения в представление **view_goods_operations** то в нем необходимо обновить набор полей при помощи редактора полей (*Fields Editor*).

Для представления **view_operation_goods_list** будем использовать компонент который назовем **Q_v_operation_goods_list**. После настройки компонентов **DataModule2** будет выглядеть примерно следующим образом (рис. 3.13):

Рис. 3.13. Добавление новых компонентов для работы с представлениями.

Для **Q_v_goods_operations** пропишем обработчик событий **AfterScroll** по аналогии с тем как это было сделано в прошлой лабораторной работе:

```
T_op_goods_list.Locate('IDoperation;IDgc',
      VarArrayOf([Q_v_operation_goods_listIDoperation.Value,
            Q_v_operation_goods_listIDgc.Value]), []
```

Функция **VarArrayOf** создает одномерный массив значений **Variant** (могут храниться данные почти любых типов данных). Для её использования необходимо чтобы в блоке **uses** в самом верху программного кода модуля был прописан заголовочный файл **System.Variants** (для старых версий **RAD Studio** просто **Variants**).

Теперь перейдем к изменению внешнего вида главной формы (Form1). Разместим на форме компонент **PageControl** у которого создадим две вкладки (рис. 3.14). Первая вкладка будет отвечать за отображение состояния склада, а вторая за перечень операции. Переместим **DBGrid1** на вкладку с операциями. Для компонента установим свойство **Align** в значение **alClient**.

Добавьте на вкладку с составом операций еще один компонент **DBGrid**, и свяжем его с **Q_v_operation_goods_list**. Далее настроим визуальное отображение. Для компонента **DBGrid2** установим свойство **Align** в значение **alBottom**.

Для того чтобы компоненты визуально не сливались, установим им свойство **AlignWithMargins** в значение **True**.

Настройте отображение столбцов как показано на (рис. 3.14).

Далее можно разместить компонент **Label** который опишет, какие данные отображаются в **DBGrid2**. Зададим ему свойство **Align** в значение **alBottom,** а свойству **Alignment** (отвечает за выравнивание текста) назначим значение **taCenter**.

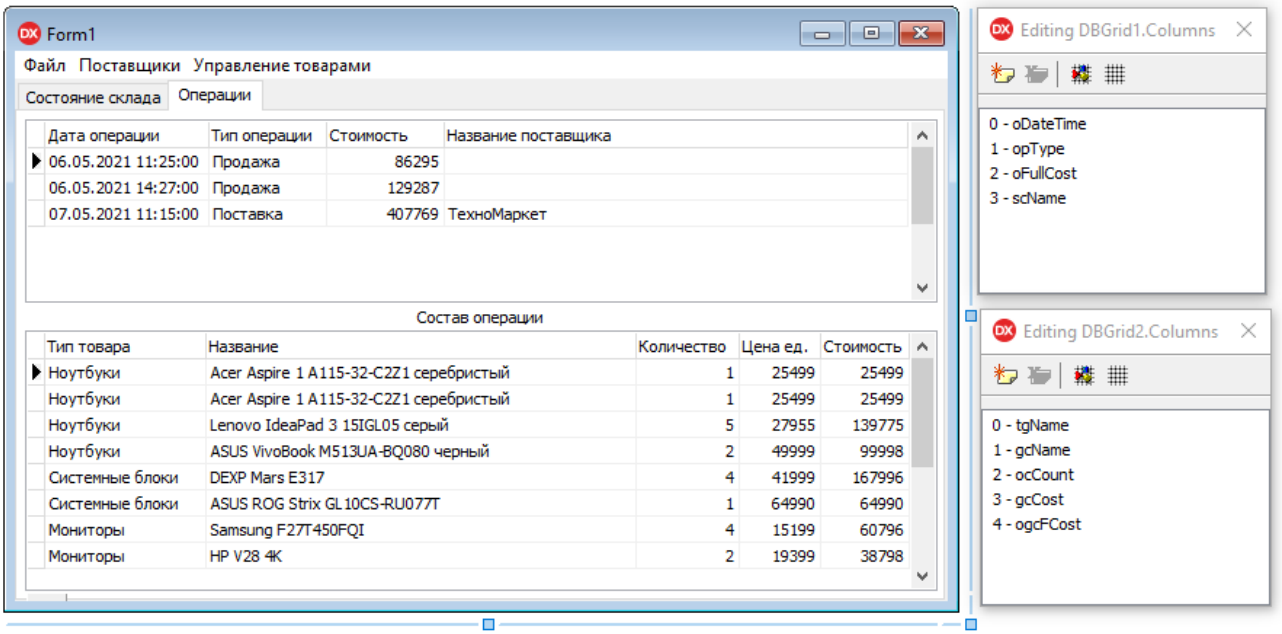

Рис. 3.14. *Form1* после настройки компонентов на вкладке **Операции**, редакторы столбцов компонентов *DBGrid1* и *DBGrid2*

Для отображения складских запасов на вкладке **Состояние склада** также разместите **DBGrid**. Свяжем его с набором данных **Q_v_storage** и зададим свойство **Align** в значение **alClient** (рис. 3.15):

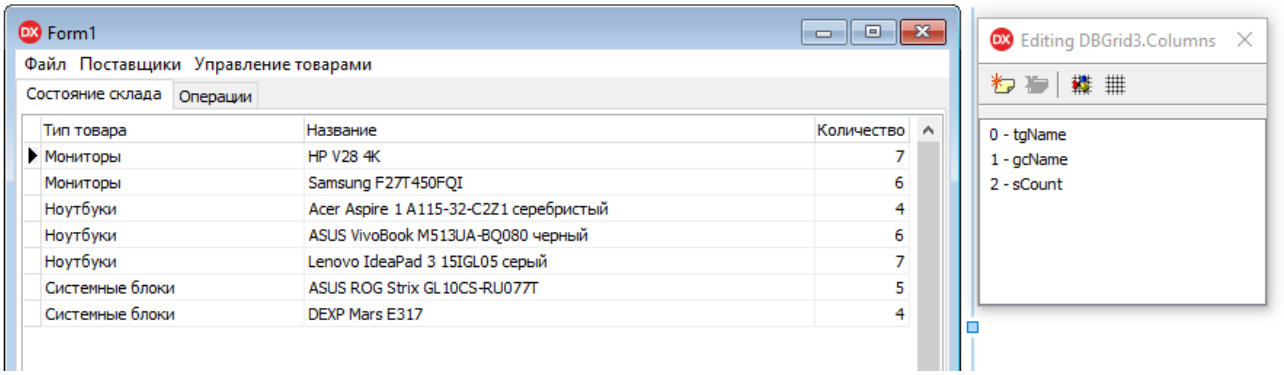

Рис. 3.15. Пример настройки *DBGrid* на вкладке **Состояние склада**

3.2 Настройка отображения данных с использованием Master–Detail

Таблицы **operations** и **op_goods_list** находятся в отношении «один-ко-многим». В таком случае для отображения данных часто применяется механизм **Master**–**Detail**, в нашем случае при выборе концертной операции из списка будем отображать данные о составе товаров, участвующих в ней.

Связь **Master**–**Detail** для компонентов **ADOTable** настраивается визуально. Настройте связь между таблицами **T_operations** и **T_op_goods_list**.

Для этого у таблицы **T_op_goods_list** установим свойство **MasterSource** в значение **DS_T_operations**.

Далее перейдем на свойство **MasterFields** и нажмем кнопку с многоточием. В появившемся окне **Field Link Designer** установим параметры связи (рис. 3.16), выбрав в обоих списках поле **IDoperation** и нажав кнопку **Add**.

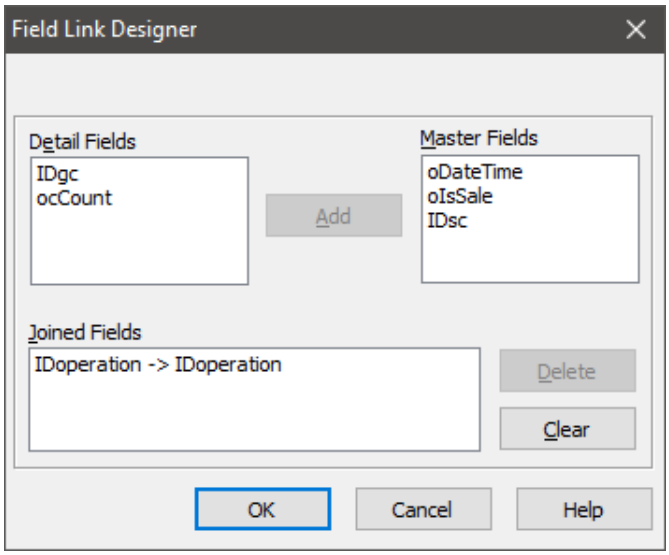

Рис. 3.16. Окно настройки соединения полей **Field Link Designer**

Поскольку для отображения информации о составе операций используется **Q_v_operation_goods_list**, то также настроим для него связь *Master–Detail* с **Q_v_goods_operations**.

Для этого нам необходимо изменить *SQL* код прописанный в компоненте **Q_v_operation_goods_list** на следующий:

```
SELECT *
FROM view_operation_goods_list
WHERE view_operation_goods_list.IDoperation = :IDoperation
```

Далее свойству **DataSource** для **Q_v_operation_goods_list** установим значение **DS_Q_v_goods_operations**, а свойству **Active** – значение **True**.

Теперь при выборе из списка какой-либо операции, будут отображаться принадлежащие ей товары (рис. 3.17):

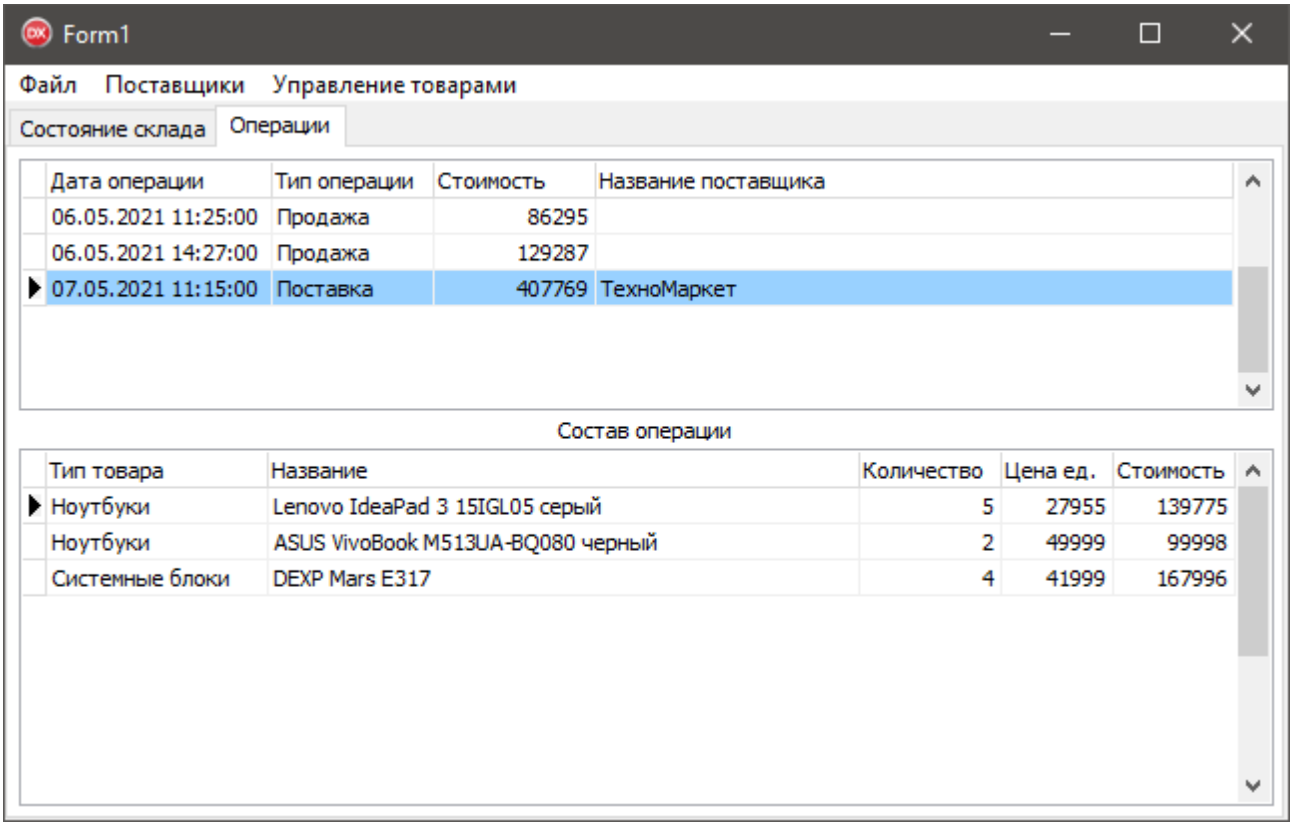

Рис. 3.17. Демонстрация работы *Master–Detail*, для отображения состава операции

Как можно заметить, в *SQL*-запросе появился блок *WHERE* в котором прописано условие, выполняющее проверку равенства, где **:IDoperation** означает параметр, значение которого берется из набора данных **T_operations** через компонент **DS_T_operations**.

3.3 Настройка выборки данных при помощи модификации SQL

При наличии большого количества данных (в нашем случае сведений о продажах и поставках) целесообразно применять выборку, которая будет давать возможность просмотреть все операции, либо только продажи или поставки.

Для этого на вкладке **Операции** разместите компонент **Panel** и задайте свойству **Align** значение **alTop**. Настройку внешнего вида панели можно осуществить при помощи свойств **BevelInner**, **BevelKind** и **BevelOuter**. Для того чтобы панель визуально слилась с формой необходимо задать значение **bvNone** для этих свойств.

Управление видом отображаемых данных будет происходить при помощи компонента **RadioGroup**. Разместите его на **Panel** и назначьте свойству **Align** значение **alLeft** (рис. 3.18):

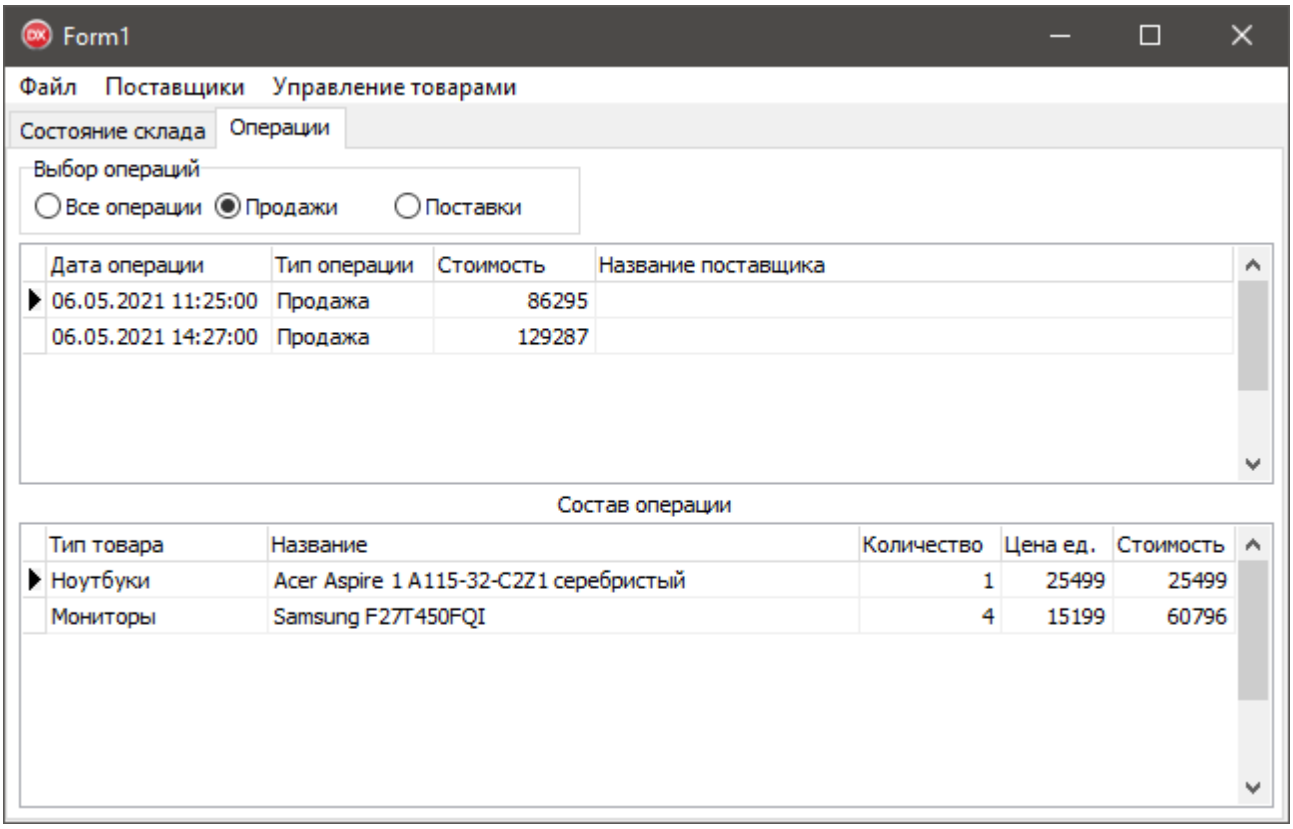

Рис. 3.18. Размещение компонента *RadioGroup*

Теперь нажав на кнопку с многоточием у свойства **Items** появится редактор строк. В каждой новой строке которого пропишите следующие значения: **Все операции**, **Продажи**, **Поставки**.

Для того чтобы элементы были расположены горизонтально необходимо задать значение **3** свойству **Columns**

Для выбора элемента по умолчанию установите свойству **ItemIndex** значение **0** (выбор на **Все операции**).

Для изменения заголовка **RadioGroup** в свойство **Caption** пропишите **Выбор операций**.

Теперь для организации выбора для **RadioGroup** пропишите код на событие **OnClick**:

```
procedure TForm1.RadioGroup1Click(Sender: TObject);
begin
    // Отключаем связь с визуальным компонентом
    DataModule2.Q_v_goods_operations.DisableControls;
    // Деактивируем НД Q_v_goods_operations
    DataModule2.Q_v_goods_operations.active := false;
    // Очищаем SQL код в НД Q_v_goods_operations
    DataModule2.Q_v_goods_operations.SQL.Clear;
```

```
    // Добавляем базовую часть SQL запроса на выбор данных
    DataModule2.Q_v_goods_operations.SQL.Add
      ('SELECT * FROM view_goods_operations ');

    // В зависимости от выбранного элемента в RadioGroup1 добавляем
    // условие выборки в SQL запрос
    if RadioGroup1.ItemIndex = 1 then
       DataModule2.Q_v_goods_operations.SQL.Add('WHERE oIsSale = 1 ');
    if RadioGroup1.ItemIndex = 2 then
       DataModule2.Q_v_goods_operations.SQL.Add('WHERE oIsSale = 0 ');

    // Активируем НД Q_v_goods_operations
    DataModule2.Q_v_goods_operations.active := true;
    // включаем связь с визуальным компонентом
    DataModule2.Q_v_goods_operations.EnableControls;
end;
```

Скомпилируйте приложение и проверьте работоспособность **RadioGroup**.

3.4 Редактирование данных с учетом механизма Master–Detail

В будущем на форме будем отображать полную стоимость товаров в операции (например, как итог в чеке). Для этого необходимо в **Q_v_operations_fullcost** прописать следующий запрос:

```
SELECT *
FROM view_operations_fullcost
WHERE view_operations_fullcost.IDoperation = :IDoperation
```

Поскольку в списке товаров, которые можно добавить в текущую операцию должны отображаться только отсутствующие в ней товары, то для компонента **Q_v_selector_goods** сделайте следующий *SQL*-запрос:

```
SELECT IDgc, sNameCost
FROM v_selector_goods
WHERE v_selector_goods.IDgc NOT IN
    (SELECT IDgc FROM view_operation_goods_list
     WHERE view_operation_goods_list.IDoperation = :IDoperation)
or v_selector_goods.IDgc = :IDgcSelected
```

Далее для **Q_v_operations_fullcost** и **Q_v_selector_goods** свойству **DataSource** задайте значение **DS_T_operations**. После чего активируйте компоненты, установив свойство **Active** в значение **True**.

Параметр **:IDgcSelected** необходим для передачи в *SQL*-запрос *ID* выбранного пользователем товара, и в случае если он не был создан автоматически, его необходимо создать через свойство **Parameters** компонента **Q_v_selector_goods** (рис. 3.19):

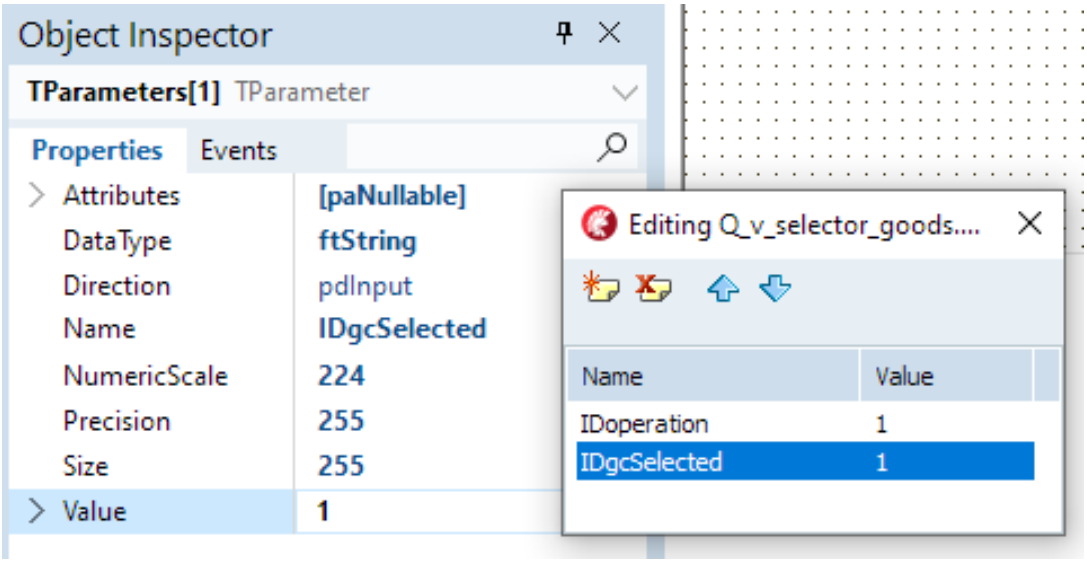

Рис. 3.19. Добавление параметров к *SQL*-запросу

Теперь необходимо снова модифицировать обработчик события **AfterScroll** компонента **Q_v_operation_goods_list** в нем задавал параметр для отображения списка не используемых в составе операции товаров. Код обработчика будет выглядеть следующим образом:

```
procedure TDataModule2.Q_v_operation_goods_listAfterScroll(DataSet: TDataSet);
begin
    if Q_v_operation_goods_list.RecordCount > 0 then
    begin
      // назначение параметра с идентификатором выбранного товара това
      Q_v_selector_goods.Parameters[1].Value :=
        Q_v_operation_goods_listIDgc.Value;
      // обновление списка для DBLookupComboBox на форме Form5
      Q_v_selector_goods.Requery();
      // выполнение синхронизации T_op_goods_list методом Locate
      T_op_goods_list.Locate('IDoperation;IDgc',
        VarArrayOf([Q_v_operation_goods_listIDoperation.Value,
        Q_v_operation_goods_listIDgc.Value]), [])
    end;
end;
```

Для добавления поставок или оформления продаж товаров будем использовать новую форму (добавьте её через меню **File**, затем **New** и выберите **VCL Form - Delphi**).

Поскольку данные об операциях с товарами хранятся в одной таблице, то на новой форме (**Form5**) предусмотрим одновременное добавление записей продаж и поставок. Различие будет заключаться в скрытии/отображении блока с указанием поставщика.

Для начала поместим панель, которая будет отвечать за настройку параметров самой операции (назовём ее **P_EditOperationInfo)**. На панели разместим два компонента **GroupBox**, которые буду отвечать за настройку параметров даты/времени операции и редактирование поставщика. Назовем компоненты **GB_DateTime** и **GB_Supplier** соответственно. В компонент **GB_DateTime** поместим компоненты **DatePicker** и **TimePicker** (рис. 3.20):

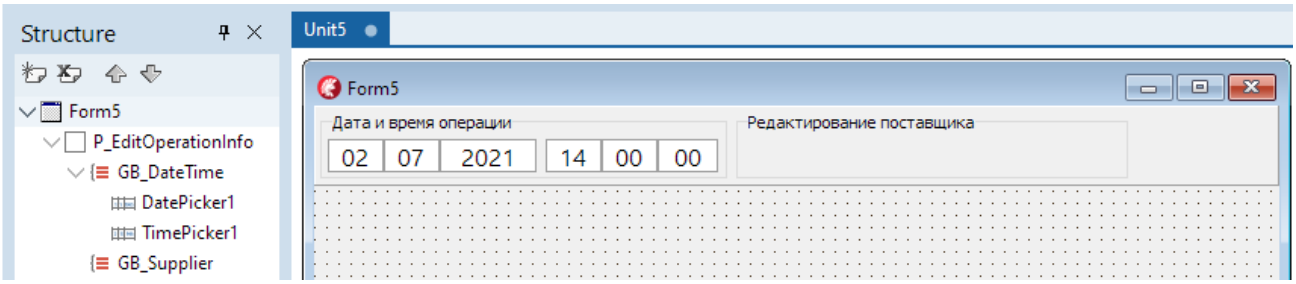

Рис. 3.20. Добавление компонентов работы с временем

Для свойства **TimeFormat** компонента **TimePicker1** зададим значение **hh:mm:ss**.

В событии **OnChange** компонента **DatePicker1** пропишем обработчик внесения изменение в поле **oDateTime:**

```
procedure TForm5.DatePicker1Change(Sender: TObject);
begin
  // если во время активности формы обнаруживаем изменение времени
  // относительно хранящегося в БД
  if Form5.Active and (DataModule2.T_operationsoDateTime.Value <>
    (DatePicker1.Date + TimePicker1.Time)) then
  begin
    // Перевод набора в режим редактирования если он не находится в данном ре
жиме
    if DataModule2.T_operations.State = dsBrowse then
      DataModule2.T_operations.edit;
    // Формирование типа TDateTime (выражается как сумма TDate и TTime)
```

```
    DataModule2.T_operationsoDateTime.Value := DatePicker1.Date +
      TimePicker1.Time;
  end;
end;
```

Такой же код прописываем для событии **OnChange** компонента **TimePicker1**.

Далее в области компонента **GB_Supplier** поместим компонент **TDBLookupComboBox** (рис. 3.21), который настроим следующим образом:

Для **DBLookupComboBox:**

Свойство	Значение
Name	DBL_supplier
DataSource	DS_T_Operations
DataField	IDsc
ListSource	DS_T_supplier_catalog
ListField	scName
KeyField	IDsc

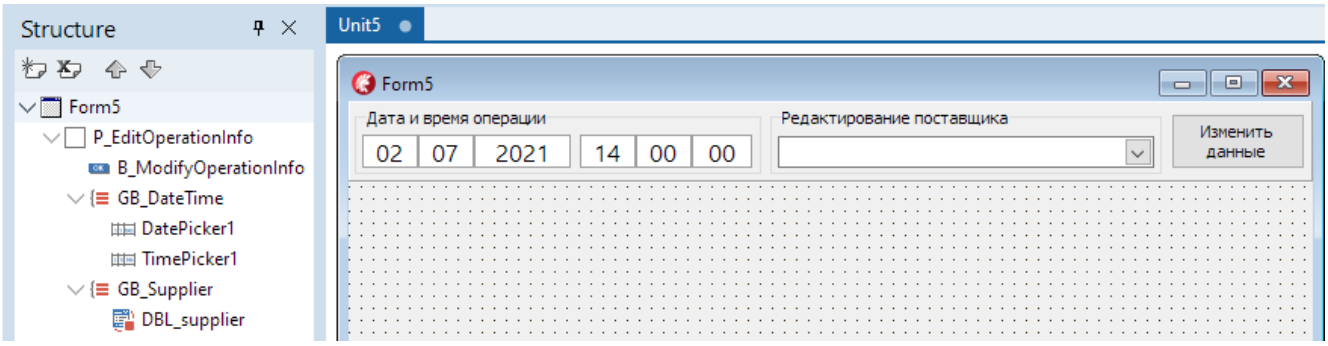

Рис. 3.21. Добавление компонентов редактирования поставщика товаров

Теперь на панель **P_EditOperationInfo** поставим кнопку и дадим ей имя **B_ModifyOperationInfo** (рис. 3.21). В обработчике **OnClick** кнопки пропишите следующий код:

```
procedure TForm5.B_ModifyOperationInfoClick(Sender: TObject);
begin
  // Если набора данных в режиме редактирования
  if DataModule2.T_operations.State = dsEdit then
    // Запоминаем изменения
    DataModule2.T_operations.post;
end;
```

Под панелью **P_EditOperationInfo** расположим **DBGrid** (рис. 3.22), который свяжем с **DS_Q_v_operation_goods_list**. В редакторе столбцов настроим поля как у **DBGrid2** на **Form1**, а также настроим выделение целой строки, переведя свойство **options (dgRowSelect)** в значение **True**.

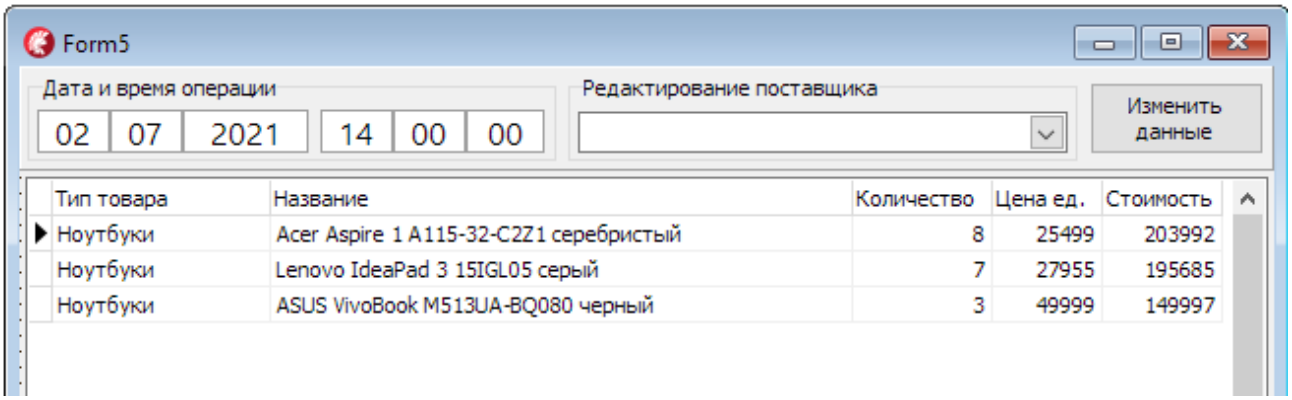

Рис. 3.22. Добавление *DBGrid* для отображения товаров

Для выполнения редактирования данных таблицы **op_goods_list** разместим на форме две панели (дадим им имена **P_BaseControl** и **P_ModifyData**). Для настройки расположение отрегулируем у компонентов параметры выравнивания (свойство **Align**). У панели **P_ModifyData** свойство **Visible** переведем в значение **False** (рис. 3.23).

Редактирование таблицы **op_goods_list** будет осуществляться по аналогии с тем, как это было в случае с редактированием каталога товаров. Разместим на панели кнопки для вызова основных действий (рис. 3.23 – 1).

Дополнительно на панели **P_BaseControl** поместим отдельную кнопку **Завершить оформление** (**B_CompleteModifyOperation**) (рис. 3.23 – 2), которая будет отвечать за завершение операции (закрытии формы), а также компонент **DBText** который будет отвечать за вывод полной стоимости товаров, участвующих в операции (рис. 3.23 – 3).

Настроим компонент **DBText1** задав свойству **DataSource** значение **DS_Q_v_operations_fullcost**, а свойству **DataField** значение **oFullCost**.

Пропишем код обработчика события **OnClick** кнопки **B_CompleteModifyOperation.**

```
procedure TForm5.B_CompleteModifyOperationClick(Sender: TObject);
begin
  // закрываем форму
  Form5.Close;
```

```
end;
```

Для изменения значения полей понадобится один компонент **DBLookupComboBox** (рис. 3.23 – 4) для выбора товара и один компонент **DBEdit** (рис. 3.23 – 5) для задания количества товаров (разместим их на панель **P_ModifyData**).

Для **DBLookupComboBox:**

Свойство	Значение
Name	DBL_goods_selector
DataSource	DS_T_op_goods_list
DataField	IDgc
ListSource	DS_Q_v_selector_goods
ListField	sNameCost
KeyField	IDgc

Теперь разместите компоненты **label** и настройте им свойство **Caption**. После настройки компонентов форма примет следующий вид.

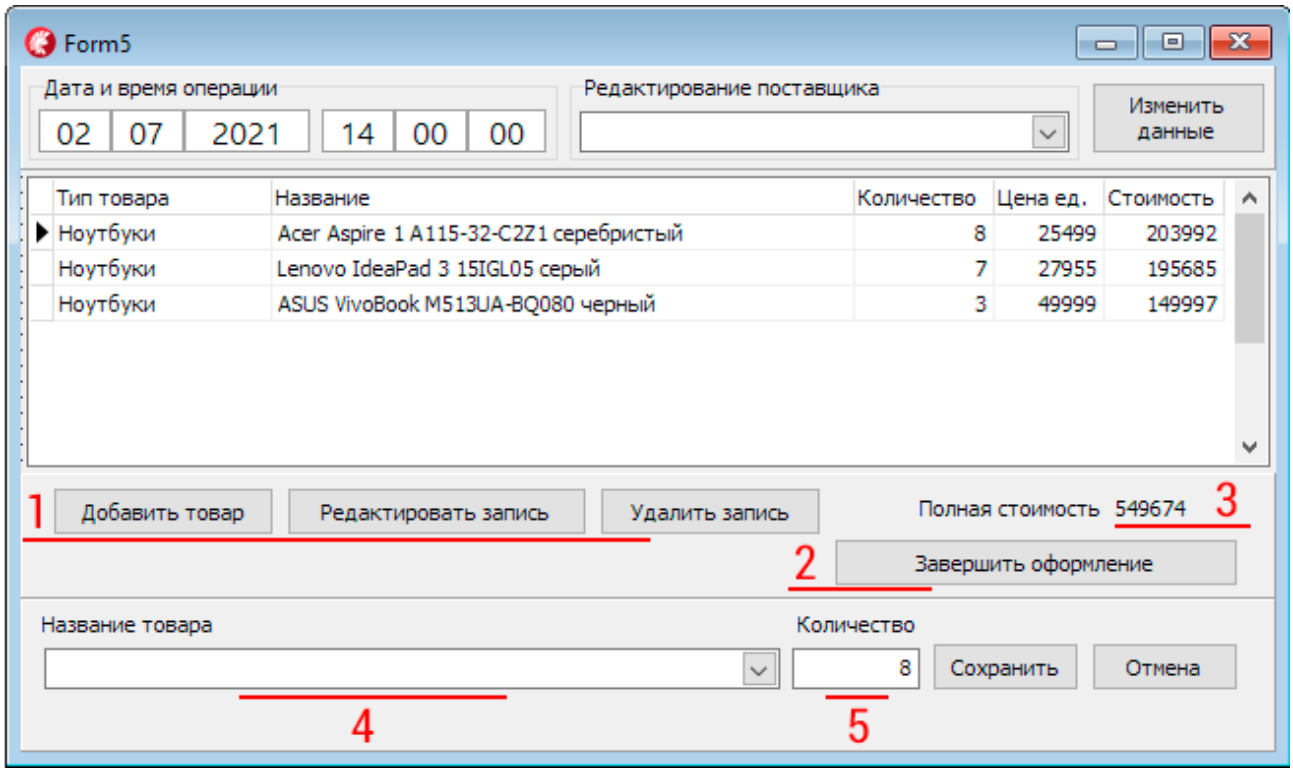

Рис. 3.23. *Form5* с необходимыми компонентами

Далее необходимо прописать все обработчики событий.

Обработчик **OnClick** кнопки **Добавить товар** (имя **B_AppendGoods**):

```
procedure TForm5.B_AppendGoodsClick(Sender: TObject);
      begin
        if DataModule2.T_op_goods_list.State = dsBrowse then
        begin
          DBGrid1.Enabled := false;
          //указываем что товар не выбран
          DataModule2.Q_v_selector_goods.Parameters[1].Value := -1;
          //обновляем селектор
          DataModule2.Q_v_selector_goods.Requery();
          // Включаем режим добавления
          DataModule2.T_op_goods_list.Append;
          P_ModifyData.Visible := true;
          P_BaseControl.Visible := false;
        end;
      end;
```

Обработчик **OnClick** кнопки **Редактировать запись** (имя **B_ModifyGoods**):

```
procedure TForm5.B_ModifyGoodsClick(Sender: TObject);
begin
  if DataModule2.T_op_goods_list.State = dsBrowse then
  begin
    DBGrid1.Enabled := false;
    //синхронизация данных со списком выбора
    DataModule2.Q_v_selector_goods.Locate('IDgc',
                                DataModule2.T_op_goods_listIDgc.Value, []);
    // активация режима редактирования
    DataModule2.T_op_goods_list.edit;
    P_ModifyData.Visible := true;
    P_BaseControl.Visible := false;
  end;
end;
```

Обработчик **OnClick** кнопки **Удалить запись** (**B_DeleteGoods**):

```
procedure TForm5.B_DeleteGoodsClick(Sender: TObject);
begin
  if DataModule2.T_op_goods_list.State = dsBrowse then
    if MessageDlg('Подтвердите удаление записи', mtConfirmation, [mbYes, mbNo],
      0) = mrYes then
      DataModule2.T_op_goods_list.Delete;
end;
```

Обработчики **OnClick** для кнопок **Сохранить** (**B_SaveData**) и **Отмена** (**B_CancelModify**):

```
procedure TForm5.B_SaveDataClick(Sender: TObject);
begin
  if DataModule2.T_op_goods_list.State in [dsEdit, dsInsert] then
  begin
    DataModule2.T_op_goods_list.post;
    P_BaseControl.Visible := true;
    P_ModifyData.Visible := false;
    DBGrid1.Enabled := true;
  end;
end;

procedure TForm5.B_CancelModifyClick(Sender: TObject);
begin
  if DataModule2.T_op_goods_list.State in [dsEdit, dsInsert] then
  begin
    DataModule2.T_op_goods_list.Cancel;
    P_BaseControl.Visible := true;
    P_ModifyData.Visible := false;
    DBGrid1.Enabled := true;
  end;
end;
```

Теперь понадобиться код, который будет обновлять данные на формах после их изменения в таблице **operations**, события **AfterPost** и **AfterDelete** :

```
procedure TDataModule2.T_operationsAfterPost(DataSet: TDataSet);
var
  IDoperation: integer;
begin
  IDoperation := T_operationsIDoperation.Value;
  Q_v_goods_operations.DisableControls();
  Q_v_goods_operations.Requery();
  Q_v_goods_operations.Locate('IDoperation', IDoperation, []);
  Q_v_goods_operations.EnableControls();
end;

procedure TDataModule2.T_operationsAfterDelete(DataSet: TDataSet);
begin
  Q_v_goods_operations.DisableControls();
  Q_v_goods_operations.Requery();
  Q_v_goods_operations.EnableControls();
end;
```

При изменении перечня товаров в составе операций нам также нужно обновить все наборы данных, которые используют данные таблицы **op_goods_list** – **Q_v_operation_goods_list** и **Q_v_operations_fullcost**. Пропишите в обработчиках событий **AfterPost** и **AfterDelete** код:

```
procedure TDataModule2.T_op_goods_listAfterDelete(DataSet: TDataSet);
begin
  Q_v_operation_goods_list.DisableControls();
  Q_v_operation_goods_list.Requery();
  Q_v_operation_goods_list.EnableControls();
  Q_v_operations_fullcost.Requery();
end;

procedure TDataModule2.T_op_goods_listAfterPost(DataSet: TDataSet);
var
  IDgcVal: integer;
begin
  IDgcVal := T_op_goods_listIDgc.Value;
  Q_v_operation_goods_list.DisableControls();
  Q_v_operation_goods_list.Requery();
  Q_v_operation_goods_list.Locate('IDgc', IDgcVal, []);
  Q_v_operation_goods_list.Requery();
  Q_v_operation_goods_list.EnableControls();
  Q_v_operations_fullcost.Requery();
end;
```

Поскольку реализован весь необходимый функционал для редактирования параметров операции и ее состава, нам остается при закрытии **Form5** сделать проверку, что все данные сохранены.

При закрытии формы возникает событие **OnCloseQuery**, которое мы будем использовать для обработки данных связанных с завершением оформления операции. Будем проверять состав товаров в операции и если количество записей о товарах будет равно нулю, то запись об операции удаляем. Также если какие-либо данные небыли сохранены, то будет предоставлена возможность сохранить их.

Обработчик события **OnCloseQuery** будет следующим:

```
procedure TForm5.FormCloseQuery(Sender: TObject; var CanClose: Boolean);
begin
  //проверка редактируем ли мы состав товаров в операции
  if P_ModifyData.Visible then
  begin
    //если мы редактируем то выводим сообщение
      ShowMessage ('В данный момент происходит изменения перечня товаров в рамках о
      перации, вы не можете завершить оформление');
    //устанавливаем для флага CanClose значение говорящее, что нельзя закрыть форму
    CanClose := false;
    //при помощи команды exit() игнорируем оставшийся код
    exit();
  end;
  //Проверка если в операции нет товаров
  if DataModule2.T_op_goods_list.RecordCount = 0 then
```

```
  begin
    //отмена изменений в таблицы операций
    if DataModule2.T_operations.State = dsEdit then
      DataModule2.T_operations.Cancel;
    //удаление записи об операции
    DataModule2.T_operations.Delete;
  end
  //В случае если в операции присутствуют товары,
  //проверяем есть ли изменения в данных
  else if DataModule2.T_operations.State = dsEdit then
  begin
    //В случае если требуется сохранить изменения,
    //запрашивается подтверждение сохранения
    if MessageDlg('Подтвердить сохранение измененных данных операции',
      mtConfirmation, [mbYes, mbNo], 0) = mrYes then
    begin
      //сохраняются изменения
      DataModule2.T_operations.post
    end
    else
      //в противном случае происходит отмена
      DataModule2.T_operations.Cancel;
  end;
  //обновляем данные в Q_v_goods_operations имитируя срабатывание события AfterPost
  DataModule2.T_operationsAfterPost(nil);
  //обновляем данные в НД Q_v_storage
  DataModule2.Q_v_storage.Requery;
end;
```

3.5 Особенности вызова формы для редактирования данных с учетом перестройкой интерфейса

Форма – **Form5** отвечает за редактирование, как состава продаж, так и поставок, содержащихся в одной таблице. Ключевым отличием является указание поставщика при поставках. Таблица **operations** содержит поле **oIsSale** определяющее тип операции – 1 означает **Продажа**, 0 – **Поставка**.

Во время вызова формы возникает событие **OnShow**. Используя данное событие мы будем скрывать или же показывать компонент с именем **GB_Supplier** для выбора поставщика, а также изменять надпись компонента **GB_DateTime** для указания типа операции.

Код обработчика будет выглядеть следующим образом:

```
procedure TForm5.FormShow(Sender: TObject);
begin
  if DataModule2.T_operationsoIsSale.Value = 1 then
  begin
    GB_Supplier.Visible := false;
```

```
    GB_DateTime.Caption := 'Дата и время продажи товара';
  end
  else
  begin
    GB_DateTime.Caption := 'Дата и время поставки товара';
    GB_Supplier.Visible := true;
  end;
end;
```

Настройка **Form5** полностью завершена, теперь перейдем на главную форму (**Form1)** и реализуем механизм вызова **Form5**.

Для этой цели мы будем использовать контекстное меню (при желании можно использовать кнопки).

Для создания контекстного меню разместите на **Form1** компонент **PopupMenu**. Настройка элементов в меню происходит аналогичным образом как в **MainMenu**. Создайте три элемента (рис. 3.24):

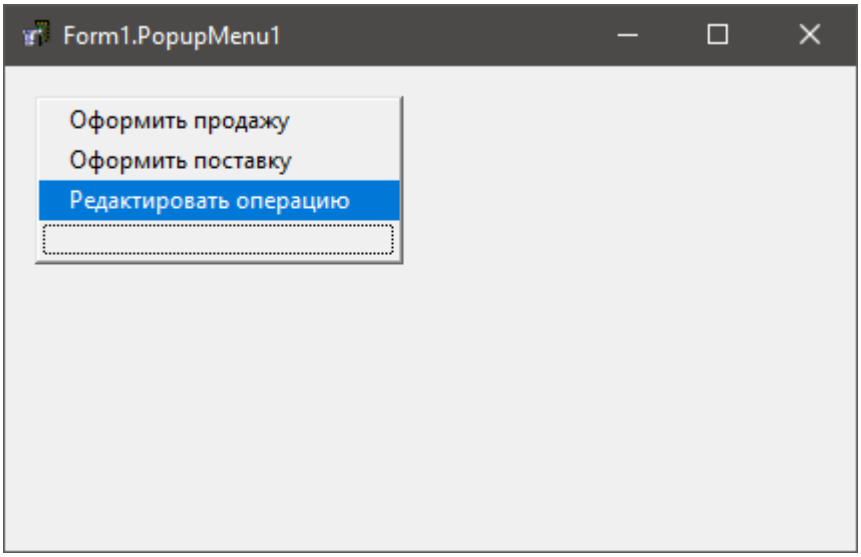

Рис. 3.24. Создание контекстного меню

Далее привяжем созданное меню к компоненту **DBGrid1** отображающему список операций.

У **DBGrid1** в свойстве **PopupMenu** укажите имя созданного контекстного меню, в нашем случае – **PopupMenu1**.

Поскольку все три элемента контекстного меню будут вызывать одну и туже форму, то для сокращения объема кода создадим процедуру **DisplayForm5**.

Процедура будет устанавливать дату и время из базы данных в компоненты **DatePicker** и **TimePicker**, а также вызывать необходимую форму.

Код процедуры **DisplayForm5**:

```
procedure DisplayForm5();
begin
  //Назначение даты (Trunc "отрезает" время)
  Form5.DatePicker1.Date := Trunc(DataModule2.T_operationsoDateTime.value);
  //Назначение времени
  Form5.TimePicker1.Time := DataModule2.T_operationsoDateTime.AsDateTime;
  //вызов показа формы
  Form5.ShowModal;
end;
```

Данный код следует разместить в начале модуля **Unit1**, после подключения других модулей (рис. 3.25):

```
unit Unit1;

interface ...

implementation

{$R *.dfm}

uses Unit2, Unit3, Unit4, Unit5;

procedure DisplayForm5();
begin
  //Назначение даты (Trunc "отрезает" время)
  Form5.DatePicker1.Date := Trunc(DataModule2.T_operationsoDateTime.value);
  //Назначение времени
  Form5.TimePicker1.Time := DataModule2.T_operationsoDateTime.AsDateTime;
  //вызов показа формы
  Form5.ShowModal;
end;
```

Рис. 3.25. Фрагмент кода модуля *Unit1*

Для событий **OnClick** элементов контекстного меню **Оформить продажу** (**PM_add_sale**), **Оформить поставку** (**PM_add_supply**) и **Редактировать операцию** (**PM_Edit_Operation**) пропишите следующие обработчики:

```
procedure TForm1.PM_add_saleClick(Sender: TObject);
begin
  // функция Now() возвращает текущие дату и время
  DataModule2.T_operations.AppendRecord([nil, Now(), 1]);
  DisplayForm5();
end;

procedure TForm1.PM_add_supplyClick(Sender: TObject);
begin
  DataModule2.T_operations.AppendRecord([nil, Now(), 0]);
  DisplayForm5();
```

```
end;

procedure TForm1.PM_Edit_OperationClick(Sender: TObject);
begin
  DisplayForm5();
end;
```

Процедура **AppendRecord** заполняет поля таблицы значениями, которые передаются через массив **Values** (в квадратных скобках). При этом первое значение в массиве присваивается первому полю, второе – второму и т. д. Естественно, что значения в массиве должны быть совместимы с полями таблицы. В случае если полю будет передано значение **nil**, оно сохранит свое текущее значение.

КОНТРОЛЬНЫЕ ВОПРОСЫ

Для чего применяется связь *Master–Detail*?

Что из себя представляет группировка данных (приведите пример сгруппированных данных)?

Опишите назначение агрегатных функций. Какие виды агрегирования поддерживает *MySQL*?

Каким образом в *MySQL* работает функция **IF**?

Для чего применяется функция **CONCAT**?

Дайте характеристику свойству **AlignWithMargins** визуальных компонентов.

Что из себя представляют параметры *SQL*-запросов (например, *:IDoperation*)?

Чем компонент **RadioGroup** отличается от **CheckListBox**?

Что описывает маска формата **hh:mm:ss**, какие еще варианты существуют?

Охарактеризуйте свойство **dgRowSelect**.

Для чего применяется процедура **Requery**().

Чем **PopupMenu** отличается от **MainMenu**?

Для чего создаются уникальные индексы. Для чего предназначены индексы?

Что из себя представляет связь многие-ко-многим, приведите пример данных с такой связью?

Как реализуется связь многие-ко-многим в реляционной базе данных?

Дайте характеристику типу данных **Variant**, для чего он применяется?