МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. Р.Е.Алексеева

Кафедра «Информатика и системы управления»

БАЗЫ ДАННЫХ

КОМПОНЕНТЫ ОТОБРАЖЕНИЯ ДАННЫХ

Методические указания к лабораторной работе №3 для студентов
специальности 230102, 230201

Нижний Новгород 2010

Составитель Балашова Т.И.

ББК

БАЗЫ ДАННЫХ. КОМПОНЕНТЫ ОТОБРАЖЕНИЯ ДАННЫХ: учебно-методические указания к лабораторной работе №3 для студентов специальности 230102, 230201/ НГТУ; сост.: Балашова Т.И. Н.Новгород 2010. 17с.

В учебно-методические материалах дано описание использования основных компонентов отображения данных на примере разработки базы данных с использованием Borland Delphi 7

Подписано к печати . Формат 60x84 1/16. Бумага офсетная.
Печать офсетная. Усл.печ.л. Уч.-изд. л. .Тираж экз. Заказ .

Нижегородский государственный технический университет
им. Р.Е. Алексеева
Типография НГТУ им. Р.Е. Алексеева
Адрес университета и полиграфического предприятия:
603950, Нижний Новгород, ул. Минина, 24.

Цель работы: изучение возможностей компонентов отображения и механизмов синхронного просмотра данных в Borland Delphi.

Краткие сведения из теории

1. Компоненты отображения данных

До этого момента мы рассмотрели аспекты создания приложений баз данных, касающиеся организации доступа к данным и создания в приложениях наборов данных. Теперь более подробно остановимся на вопросах отображения данных в приложениях (интерфейс приложений).

Отображение данных обеспечивает достаточно представительный набор компонентов VCL Delphi. Для связи с набором данных эти компоненты используют компонент TDataSource.

Механизмы управления данными реализованы в компонентах наборов данных и активно взаимодействуют с компонентами отображения данных.

Рассмотрим подробнее следующие вопросы:

- использование стандартных компонентов отображения данных;
- навигация по данным;
- механизм синхронного просмотра данных;
- использование графиков для представления данных.

1.1. Классификация компонентов отображения данных

Все компоненты отображения данных можно разделить на группы по нескольким критериям (рис.1).

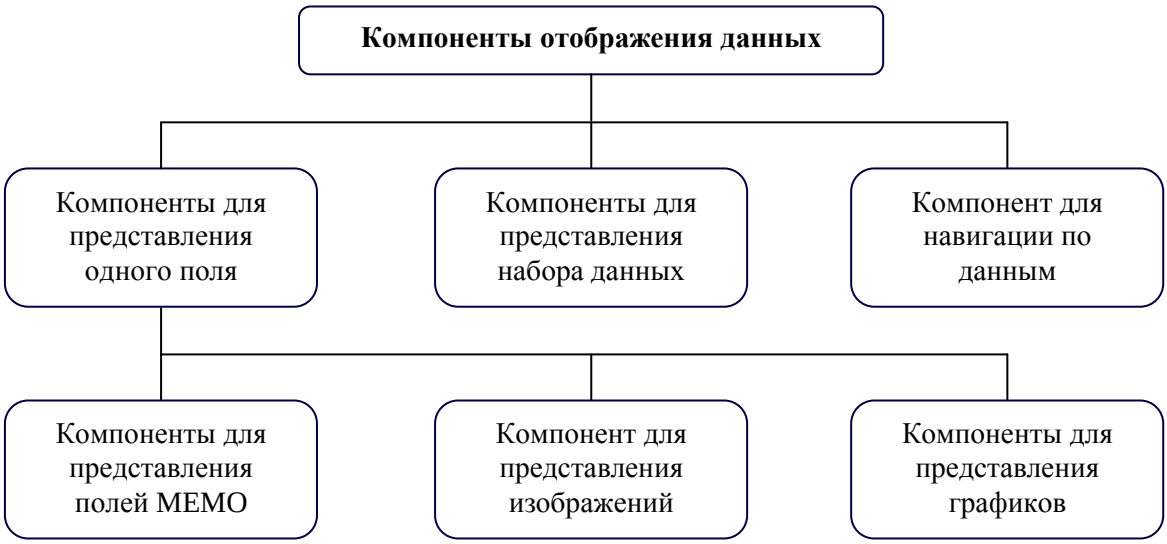

Рис. 1. Классификация компонентов отображения данных

Большинство компонентов предназначены для работы с отдельным полем, т. е. при перемещении по записям набора данных такие компоненты показывают текущие значения только одного поля. Для соединения с набором данных через компонент TDataSource предназначено свойство DataSource. Поле задается свойством DataField.

Компоненты TDBGrid и TDBCtrlGrid обеспечивают просмотр наборов данных целиком или в произвольном сочетании полей. В них присутствует только свойство DataSource.

Особенную роль среди компонентов отображения данных играет компонент TDBNavigator. Он не показывает данные и не предназначен для их редактирования, зато обеспечивает навигацию по набору данных.

Наиболее часто в практике программирования используются компоненты TDBGrid, TDBEdit и TDBNavigator.

Для представления и редактирования информации, содержащейся в полях типа Memo, используются специальные компоненты TDBMemo и TDBRichEdit. Для просмотра (без редактирования) изображений предназначен компонент TDBImage. Отдельную группу составляют компоненты синхронного просмотра данных. Они обеспечивают показ значений поля из одной таблицы в соответствии со значениями поля из другой таблицы. Наконец, данные можно представить в виде графика. Для этого предназначен компонент TDBChart. Как видно из приведенной классификации, набор компонентов отображения данных весьма разнообразен и позволяет решать задачи по созданию любых интерфейсов для приложений баз данных. Ввиду общности решаемых задач, компоненты отображения данных имеют несколько важных общих свойств, которые представлены в табл. 1 и в дальнейшем изложении опущены.

Таблица 1.

Свойство	**Описание**
property DataField: string;	Поле связанного с компонентом набора данных
property DataSource: TDataSource;	Связываемый с компонентом компонент TDataSource
property Field: Tfield;	Обеспечивает доступ к классу TField, который соответствует полю набора данных, заданному свойством DataField
property Readonly: Boolean;	Управляет работой режима "только для чтения"

1.2. Табличное представление данных

1.2.1. Компонент TDBGrid

Этот компонент используется для отображения двумерной таблицы, в которой строки представляют собой записи, а столбцы — поля набора данных.

В компоненте TDBGrid можно отображать произвольное подмножество полей используемого набора данных, но число записей ограничить нельзя — в компоненте всегда присутствуют все записи связанного набора данных. Требуемый набор полей можно составить при помощи специального Редактора столбцов (рис. 2), который открывается при двойном щелчке на компоненте, перенесенном на форму, или кнопкой свойства «Columns» в Инспекторе объектов.

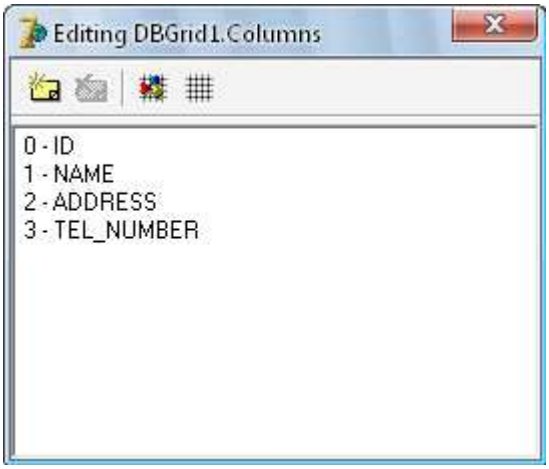

Рис. 2. Редактор столбцов

Новая колонка добавляется при помощи кнопки **Add New**, после этого ее название появляется в списке колонок. Для выбранной в списке колонки доступные для редактирования свойства появляются в Инспекторе объектов. Колонки в списке можно редактировать, удалять, менять местами.

При помощи кнопки **Add All Fields** в сетку можно добавить все поля набора данных.

Настройка параметров компонента TDBGrid, от которых зависит его внешний вид и некоторые функции, осуществляется при помощи свойства options. Текущая позиция в двумерной структуре данных может быть определена свойствами SelectedField, SelectedRows, Selectedlndex.

При необходимости разработчик может использовать разнообразные методы – обработчики событий. Среди них есть как стандартные методы, присущие всем элементам управления, так и специфические.

Например, при помощи метода – обработчика OnEditButtonClick можно предусмотреть вызов специализированной формы при щелчке на кнопке в ячейке:

```
procedure TGridForm.DBGrid1EditButtonClick(Sender: TObject);
begin
if DBGrid1.Selectedindex = 1 then TestFormUnit.ShowModal;
end;
```

Примечание: *Объект колонки в редакторе колонок имеет свойство ButtonStyle. Если ему присвоить значение cbsEllipsis, то при активизации ячейки этой колонки в правой части ячейки появляется кнопка.*

Одним из удобств, при использовании данного компонента, является возможность придать каждой колонке список, который разворачивается при щелчке на кнопке в активной ячейке колонки. Выбранное в списке значение автоматически заносится в ячейку. Для реализации этой возможности применяется свойство PickList в редакторе колонок типа TStrings. Достаточно лишь заполнить список значениями во время разработки или выполнения.

Также в Delphi доступна возможность сохранения всех данных из существующих колонок в файле или потоке при помощи методов SaveToFile и SaveToStream, а затем загрузить их обратно методами LoadFromFile И LoadFromStream.

Для этого поместим кнопку в ячейках некоторого столбца, например номер три (отсчет столбцов начинается с 0). После этого в процедуре *DBGridEditButtonClick* поместим следующий код:

```
if DBGrid1.SelectedIndex = 3 then
  begin
  DBGrid1.Columns.SaveToFile('stream.dat');
  ShowMessage('Save Complete');
  end
```

Теперь, при запуске приложения и нажатии на кнопке в любой из ячеек столбца номер три, будет появляться окно сообщения «Save Complete», а также файл “Stream.dat” в каталоге с программой.

Примечание*: одновременное использование кнопки и выпадающего списка в ячейке невозможно. При этом будет вызываться процедура, определенная под кнопкой. Пример использования DBGrid продемонстрирован в папке DBGrid на компакт – диске.*

1.3. Навигация по набору данных

Перемещение, или навигация по записям набора данных, может осуществляться несколькими путями. Например, в компонентах TDBGrid и TDBCtrlGrid, которые отображают сразу несколько записей набора данных, можно использовать клавиши вертикального перемещения курсора или вертикальную полосу прокрутки.

Но что делать, если на форме находятся только компоненты, отображающие одно поле только текущей записи набора данных (TDBEdit, TDBComboBox и т. д.)? Очевидно, что в этом случае на форме должны быть расположены дополнительные элементы управления, отвечающие за перемещение по записям.

Аналогично ни один компонент отображения данных не имеет встроенных средств для создания и удаления записей целиком.

Для решения указанных задач и предназначен компонент TDBNavigator (рис. 3), который представляет собой совокупность управляющих кнопок и выполняет операции навигации по набору данных, а также, модификации записей целиком.

Компонент TDBNavigator при помощи свойства DataSource связывается с компонентом TDataSource и через него с набором данных. Такая схема позволяет обеспечить изменение текущих значений полей сразу во всех связанных с TDataSource компонентах отображения данных. Таким образом, TDBNavigator только дает команду на выполнение перемещения по набору данных или другой управляющей операции, а всю реальную работу выполняют компонент набора данных и компонент TDataSource. Компонентам отображения данных остается только принять новые данные от своих полей.

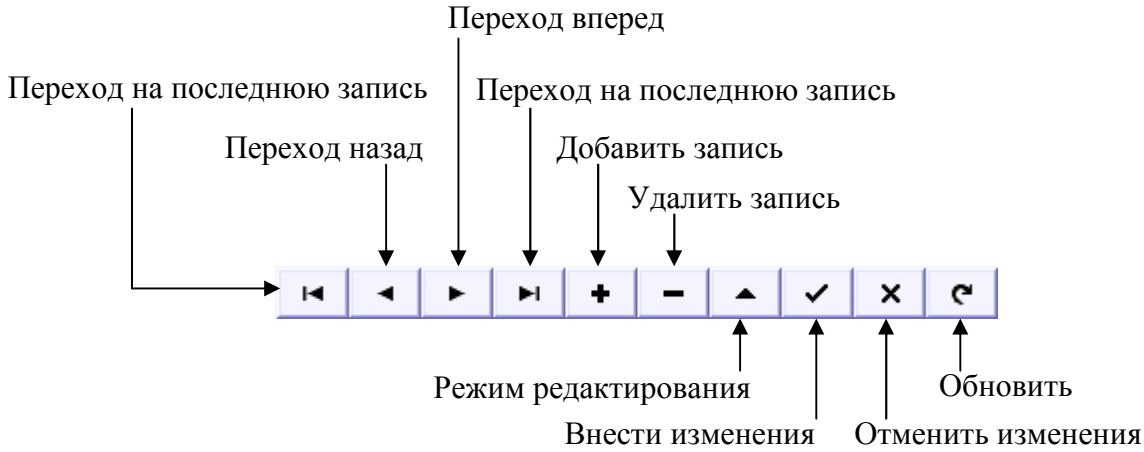

Рис. 3. DBNavigator

Самой распространенной ошибкой, ведущей к возможной потере данных, является операция случайного удаления записи, поэтому при помощи свойства «Confirmdelete» можно включить механизм контроля удаления. При каждом удалении записи нужно будет дать подтверждение выполняемой операции.

Нажатие любой кнопки можно эмулировать программно, при помощи метода BtnClick.

В случае необходимости выполнения дополнительных действий при щелчке на любой кнопке можно воспользоваться обработчиками событий BeforeAction и OnClick, в которых параметр Button определяет нажатую кнопку.

Примечание: *пример использования DBNavigator продемонстрирован на компакт – диске в каталоге DBNabigator.*

1.4. Представление отдельных полей

Большинство компонентов отображения данных предназначено для представления данных из отдельных полей. Для этого все они имеют свойство DataField, которое указывает на требуемое поле набора данных.

В зависимости от типа данных поля могут использовать различные компоненты. Для большинства стандартных полей используются компоненты TDBText, TDBEdit, TDBComboBox, TDBListBox.

Данные в формате Memo отображаются компонентами TDBMemo и TDBRichEdit. Для показа изображений предназначен компонент TDBImage.

1.4.1. Компонент TDBText

Этот компонент представляет собой статический текст, который отображает текущее значение некоторого поля связанного набора данных. При этом данные можно просматривать в режиме «Read Only» (только для чтения).

При использовании компонента следует обратить внимание на возможную длину отображаемых данных. Для предотвращения обрезания текста можно использовать свойства AutoSize и Wordwrap.

1.4.2. Компонент TDBEdit

Компонент представляет собой стандартный однострочный текстовый редактор, в котором отображаются и изменяются данные из поля связанного набора данных.

Компонент может осуществлять проверку редактируемых данных по заданной для поля маске. Саму же маску можно задать в связанном с редактором поле. Объект TField имеет собственное свойство EditMask, которое и используется при проверке данных в редакторе.

Проверка редактируемого текста на соответствие маске осуществляется методом ValidateEdit после каждого введенного или измененного символа. В случае ошибки генерируется исключение ValidateError и курсор устанавливается на первый ошибочный символ.

1.4.3. Компонент TDBCheckBox

Компонент представляет собой почти полный аналог обычного флажка (компонент TCheckBox) и предназначен для отображения и редактирования любых данных, которые могут иметь только два значения.

Предопределенные значения задаются свойствами Valuechecked и ValueUnchecked. По умолчанию они имеют значения True и False. Этим свойствам можно также присваивать любые строковые значения, причем одному свойству можно назначить несколько возможных значений, разделенных точкой с запятой.

Включение флажка происходит, если значение поля набора данных совпадает со значением свойства Valuechecked (единственным или любым из списка). Если же флажок включил пользователь, то значение поля данных приравнивается к единственному или первому в списке значению свойства ValueChecked.

Аналогичные действия происходят и со свойством ValueUnchecked.

1.4.4. Компонент TDBRadioGroup

Компонент представляет собой стандартную группу переключателей, состояние которых зависит от значений поля связанного набора данных. В поле можно передавать фиксированные значения, связанные с отдельными переключателями в группе.

Если текущее значение связанного поля соответствует значению какого-либо переключателя, то он включается. Если пользователь включает другой переключатель, то связанное с переключателем значение заносится в поле. Возможные значения, на которые должны реагировать переключатели в группе, заносятся в свойство Values при помощи специального редактора в Инспекторе объектов или программно посредством методов класса TStrings. Каждому элементу свойства Values соответствует один переключатель (порядок следования сохраняется).

Свойство Items содержит список поясняющих надписей для переключателей группы. Если для какого – либо переключателя нет заданного значения, но есть поясняющий текст, то такой переключатель включается при совпадении значения связанного поля с поясняющим текстом.

Текущее значение связанного поля содержится в поле Value.

1.4.5. Компонент TDBListBox

Компонент отображает текущее значение связанного с ним поля набора данных и позволяет изменить его на любое фиксированное из списка. Функционально компонент ничем не отличается от компонента TListBox. Значение поля должно совпадать с одним из элементов списка.

1.4.6. Компонент TDBComboBox

Компонент отображает текущее значение связанного с ним поля набора данных в строке редактирования, при этом значение поля должно совпадать с одним из элементов разворачивающегося списка. Текущее значение можно изменить на любое фиксированное из списка компонента.

Компонент может работать в пяти различных стилях, которые определяются свойством Style.

Специальных методов компонент не содержит.

1.4.7. Компонент TDBMemo

Компонент представляет собой обычное поле редактирования, к которому подключается поле с типом данных Memo или BLOB. Основное его преимущество — возможность одновременного просмотра и редактирования нескольких строк переменной длины. Компонент может отображать только строки, которые целиком видны по высоте.

В компоненте можно использовать буфер обмена при помощи стандартных средств операционной системы или унаследованными от предка TCustomMemo методами CopyToClipBoard, CutToClipBoard, PasteFromClipBoard.

Для ускорения навигации по набору данных при отображении полей типа BLOB можно использовать свойство AutoDisplay. При значении True любое новое значение поля автоматически отображается в компоненте. При значении False новое значение появляется только после двойного щелчка на компоненте или после нажатия клавиши «Enter» при активном компоненте.

Метод LoadMemo используется автоматически при загрузке значения поля, если свойство AutoDispiay = False.

Поведением компонента при работе со слишком длинными строками можно управлять при помощи свойства Wordwrap. При значении True слишком длинная строка сдвигается влево при перемещении текстового курсора за правую границу компонента. При значении False остаток длинной строки переносится на новую строку, при этом реально новая строка в данных не создается.

1.4.8. Компонент TDBImage

Компонент предназначен для просмотра изображений, хранящихся в базах данных в графическом формате.

Редактировать изображения можно только в каком-либо графическом редакторе, перенося исходное и измененное изображение при помощи буфера обмена. Это делается средствами операционной системы пользователем или программно при помощи методов CopyToClipboard, CutToClipboard, PasteFromClipboard.

Визуализация изображения осуществляется при помощи свойства Picture, которое представляет собой экземпляр класса TPicture.

Также можно полностью заменить существующее изображение или сохранить новое в новой записи набора данных. Для этого используются методы свойства Picture.

1.4.9. Компонент TDBRichEdit

Компонент предоставляет возможности полноценного текстового редактора для просмотра и изменения текстовых данных, хранящихся в связанном поле набора данных. Поле должно содержать информацию о форматировании текста.

Внешне компонент ничем не отличается от поля редактирования, поэтому о реализации доступа к гораздо более богатым возможностям редактора через интерфейс пользователя должен позаботиться разработчик. Для этого можно использовать дополнительные элементы управления.

2. Синхронный просмотр данных

При разработке приложений для работы с базами данных часто возникает необходимость в связывании двух наборов данных по ключевому полю. Например, в таблице Orders (содержит данные о заказах) имеется поле CustNo, которое содержит идентификационный номер покупателя. Под этим

же номером в таблице Customers хранится информация о покупателе (адрес, телефон, реквизиты и т. д.). При разработке пользовательского интерфейса приложения баз данных необходимо, чтобы при просмотре перечня заказов в форме приложения отображались не идентификационные номера покупателей, а их параметры.

Таким образом, в наборе данных заказов вместо поля номера покупателя должно появиться поле имени покупателя из таблицы Customers. Механизм связывания полей из различных наборов данных по ключевому полю называется синхронным просмотром. В рассмотренном примере ключевым является поле CustNo из таблицы Customers, а выбор конкретного наименования производится по совпадению значений ключевого поля и заменяемого поля из исходного набора данных — Orders. Причем необходимо, чтобы в таблице Customers поле CustNo было уникальным (составляло первичный или вторичный ключ).

Помимо простого синхронного просмотра данных, может возникнуть задача редактирования данных в аналогичной ситуации. Для этого предназначены специальные компоненты синхронного просмотра данных, которые позволяют, например, выбирать покупателя из списка, а изменится при этом номер покупателя в наборе данных заказов. Использование таких компонентов делает пользовательский интерфейс значительно более удобным и наглядным. В VLC Delphi есть два таких компонента: TDBLookupListBox и TDBLookupComboBox.

***Примечание**: На странице Win 3.1 Палитры компонентов имеются еще два компонента: TDBLokupList и TDBLookupCombo. Они обладают тем же набором функций, используются для обеспечения совместимости с приложениями, созданными в среде разработки Delphi 1, и поэтому здесь не рассматриваются.*

3. Механизм синхронного просмотра

Как и в любом другом компоненте отображения данных, в компонентах синхронного просмотра должны присутствовать средства связывания с требуемым полем некоторого набора данных. Это уже известные свойства: DataSource — применяется для задания набора данных через компонент TDataSource, и DataField — для определения требуемого поля набора данных.

Теперь необходимо задать таблицу синхронного просмотра, ключевое поле и поле синхронного просмотра.

Набор данных, содержащий указанные поля, определяется через соответствующий компонент TDataSource в свойстве ListSource.

Ключевое поле задается свойством KeyField. Во время работы компонента в свойстве KeyValue содержится текущее значение, которое связывает между собой два набора данных.

Поле синхронного просмотра определяется свойством ListField. Здесь можно задавать сразу несколько полей, которые будут отображаться в компоненте синхронного просмотра. Названия полей разделяются точкой с запятой. Если свойство не определено, то в компоненте будут отображаться значения ключевого поля.

3.1. Компонент TDBLookupListBox

Компонент представляет собой список значений поля синхронного просмотра для поля, заданного свойством DataField, из набора данных DataSource. Его основное назначение — автоматически устанавливать соответствие между полями двух наборов данных по одинаковому значению заданного поля исходной таблицы и ключевого поля таблицы синхронного просмотра. В списке синхронного просмотра отображаются возможные значения для редактирования поля основной таблицы.

3.2. Компонент TDBLookupComboBox

Компонент представляет собой комбинированный список значений поля синхронного просмотра для поля, заданного свойством DataField, из набора данных DataSource. Его основное назначение — автоматически устанавливать соответствие между полями двух наборов данных по одинаковому значению заданного поля исходной таблицы и ключевого поля таблицы синхронного просмотра. В списке синхронного просмотра отображаются возможные значения для редактирования поля основной таблицы.

Примечание: *пример использования механизма синхронного просмотра продемонстрирован на компакт – диске в каталоге DBLookUpCombo.*

4. Графическое представление данных

Для представления данных из некоторого набора данных в виде графиков различных видов предназначен компонент DBChart (Рис. 4). В нем можно одновременно показывать графики для нескольких полей данных. Графики строятся на основе всех имеющихся в наборе данных значений полей. Функционально компонент ничем не отличается от компонента Chart.

Настройка параметров компонента осуществляется специальным редактором, который можно открыть двойным щелчком на перенесенном на форму компоненте.

Редактор имеет две главные страницы — Chart и Series. Страница Chart содержит многостраничный блокнот и предназначена для настройки параметров самого графика. Страница Series также содержит многостраничный блокнот и используется для настройки серий значений данных и является основой любого графика в компоненте DBChart. Для того чтобы построить график значений некоторого поля набора данных, необходимо выполнить следующие действия, большинство из которых выполняется в специализированном редакторе компонента:

1. Создать новую серию и определить ее тип.
2. Задать для серии набор данных.
3. Связать с осями координат нужные поля набора данных и, в зависимости от типа серии, задать дополнительные параметры.
4. Открыть набор данных.

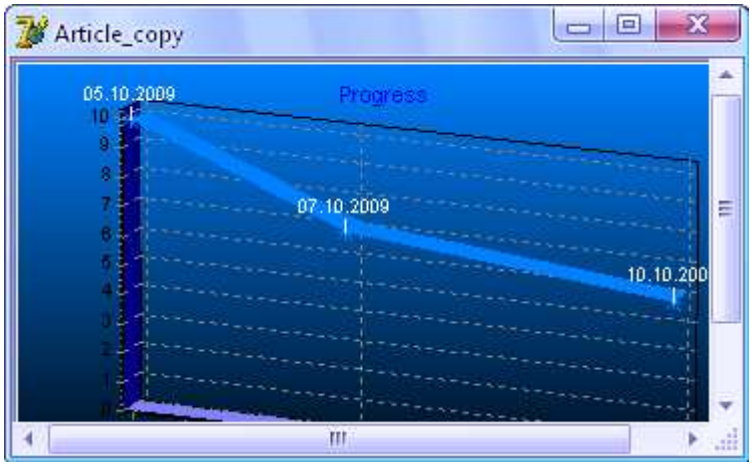

Рис. 4. DBChart

Для создания новой серии необходимо в редакторе перейти на главную страницу Chart, а на ней открыть страницу Series. На этой странице нужно щелкнуть на кнопке Add, а затем в появившемся диалоге выбрать тип серии. После этого в списке на странице Series появляется строка новой серии. Здесь можно переопределить тип, цвет и видимость серии, щелкнув на соответствующей зоне строки.

Все остальные страницы блокнота на главной странице Chart предназначены для настройки параметров графика.

Теперь необходимо перейти на главную страницу Series и на ней из списка названий серий выбрать необходимую. После этого на странице Data Source из списка выбирается строка DataSet. Далее в появившемся списке DataSet выбирается нужный набор данных.

Список X позволяет выбрать поле набора данных, значения которого будут последовательно откладываться по оси абсцисс. Список Y позволяет

выбрать поле набора данных, значения которого будут отложены по оси ординат. Соответствие между значениями полей по двум осям определяется принадлежностью к одной записи набора данных. Выбор поля в списке Labels привязывает его значения в виде меток к оси абсцисс.

Пример практической реализации компонентов отображения данных

Рассмотрим, как перечисленные выше компоненты можно использовать для создания проекта базы данных. В качестве примера будем использовать рассмотренную в предыдущих лабораторных работах базу данных сети продуктовых магазинов.

5.1. Создание форм

Для создания формы можно воспользоваться меню File → New → Form или пиктограммой на стандартной панели.

После того как форма создана, необходимо определить тип данных, которые она будет отображать (поля таблицы БД). Если это будет форма, отображающая поля таблицы – словаря, то на нее необходимо поместить несколько компонентов DBEdit. Количество компонентов зависит от того, сколько информационных полей содержится в таблице. Например, для таблицы «Article Type» (тип товара) это количество равно единице, так как поле кода записи не будет нести никакой информационной нагрузки для пользователя разрабатываемой БД и поэтому должно оставаться скрытым. Для связи компонента с полем таблицы необходимо в свойстве DataSource этого компонента указать источник данных типа TDataSource (из выпадающего списка). Если для размещения компонентов доступа, к которым относится DataSource, вы использовали DataModule, то в свойстве DataSource необходимо вручную прописать имя модуля и, через точку, имя компонента типа TDataSource, связанного с необходимой таблицей. Также, установим значение поля DataField из выпадающего списка на имя необходимого поля. Если свойство Active компонента TTable, связанного с компонентом TDataSource, было установлено в положение True, то, при соеденении поля DBEdit с полем таблицы, в нем отобразится значение этого поля для первой записи.

Добавив необходимое количество полей, поместим на форму компонент DBNavigator для работы с записями данной таблицы. Установим свойство DataSource аналогично DBEdit.

Таким образом, создадим все формы, отображающие таблицы – справочники.

Для форм, отображающих данные из таблиц, с которыми связаны данные из подчиненных таблиц, будем использовать все те же DBEdit и DBNavigator, а для связанного поля будем использовать компонент DBLookUpComboBox (рис. 5).

Для DBLookUpComboBox укажем свойство DataSource как имя главной таблицы, т.е. той, для которой создавалась форма, а в поле DataField имя того поля, которое соединено с другой таблицей. Затем укажем свойство ListSource как имя подчиненной таблицы. Свойство ListFiled установим как имя поля, которое будет отображаться в списке (может содержать имена нескольких полей), а свойство KeyField как имя ключевого поля подчиненной таблицы.

После этого, в списке будут отображаться значения поля ListSource, которые будут более информативными для конечного пользователя БД, нежели код, работа с которыми происходит внутри БД, т.е. при добавлении новой записи в связанное поле главной таблицы будет добавлен ключ записи из подчиненной.

Аналогично создаем все формы, работающие со связанными таблицами.

Рис. 5. Форма с использованием DBLookUpComboBox

После создания всех форм, необходимо реализовать переход от одной формы к другой, для редактирования записей связанной с ней таблицы. Это удобно сделать, поместив кнопку рядом с полем списка, который формируется на основе записей связанной таблицы. Кнопка должна вызывать ту форму, которая отвечает за редактирование записей таблицы, с которой связан список. Для этого добавим в процедуру ButtonClick следующий код:

```
Имя_Формы.ShowModal;
```

При нажатии на кнопку, появится форма «Имя_Формы». Причем работу с исходной формой, с которой она была «вызвана», можно продолжить только тогда, когда будет закрыта «вызванная» форма.

Иногда, для отображения статистических данных БД, удобно их графическое представление в виде различных графиков и диаграмм. Такие

возможности предоставляет компонент DBChart. Поместим его на форму. Настроим его свойства согласно описанию выше, а также, просмотрим остальные свойства, не упомянутые в описании, и настроим стиль оформления. После того, как все сделано, необходимо решить вопрос с его отображением на экране. Добавим на форму приложения кнопку. Создадим новую форму и разместим на ней график. В процедуру ButtonClick кнопки добавим:

```
chrt.Visible:=True; /chrt – имя формы графика
```

Но при увеличении формы размеры графика будут оставаться все такими же маленькими, как и исходный размер. Для того чтобы размер графика менялся пропорционально размерам формы и занимал все ее пространство, в процедуру Form Resize поместим следующий код:

```
DBChart1.Height:=Article_copy_form.Height;
DBChart1.Width:=Article_copy_form.Width;
DBChart1.RefreshData; /Обновление данных графика
DBChart1.Update;
```

После этого, при изменении размеров формы, график будет «прилипать» к ее краям.

Примечание: *пример использования компонентов отображения данных продемонстрирован на компакт диске в каталоге DBVisual.*

Задание:

На основе данных предыдущих лабораторных работ, создать графический интерфейс базы данных с использованием механизма синхронного просмотра. Представить статистические данные в виде диаграмм.

Контрольные вопросы:

1. Возможности компонента DBNavigator.
2. Какие компоненты используются для табличного представления данных.
3. Компоненты доступа к данным.
4. Как можно сохранить все данные из таблицы в потоке?
5. Для чего нужен компонент DBRadioGroup?
6. Механизм синхронного просмотра. Перечислите все элементы для работы в режиме синхронного просмотра. Как отразится на работе в режиме синхронного просмотра наличие связей между таблицами БД и почему?
7. Как организовать «прилипание» графика к краям формы при изменении ее размеров?