Уважаемые студенты групп!

Вашему вниманию представлена лекция на тему «**Компоненты для работы с базами данных».**

Задание

1. Прочитать внимательно лекцию.
2. Законспектировать лекцию в рабочую тетрадь не менее 3-6 страницы рукописного текста. В конспекте лекции обязательно должно быть приведены примеры.
3. Ответить письменно в рабочей тетради на контрольные вопросы.
4. Дата предоставления полного фотоотчета будет сообщена дополнительно.

С уважением Ганзенко Ирина Владимировна

!!! Если возникнут вопросы обращаться по телефону 0721134803 (вацап), +79591134803 (телеграмм)

disobuch.ganzenko2020@mail.ru

Лекция Компоненты для работы с базами данных(6 часов)

План

1. Доступ к данным
2. Таблица DB Grid
3. Навигация по таблице данных
4. Представление отдельных полей данных
5. Компонент DBCtrlGrid
6. Связывание данных в таблицах
7. Компоненты синхронного просмотра
8. Модуль хранения компонентов данных

1 Доступ к данным

До настоящего момента мы говорили лишь о том, как получить некий абстрактный доступ к данным из приложения, опуская самую главную, с точки зрения конечного потребителя приложения БД, возможность - собственно представление данных в приложении. Для этих целей в VCL предусмотрено 2 группы компонентов - Data Access и Data Controls.

Для доступа к данным, представленным при помощи различных компонент - будь то BDE-ориентированные источники (например, Table), или

ADO, IB Express, или dbExpress, используется один и тот же набор компонентов, расположенных на закладке Data Access:

- DataSource - источник данных;
- ClientDataSet - клиентский набор данных;
- DataSetProvider - провайдер набора данных;
- XMLTransform - преобразователь данных, представленных в виде XML в обычный пакет данных и обратно;
- XMLTransformProvider - провайдер данных для XML-документов, осуществляющий так же их обновление;
- XMLTransformClient - адаптер между XML-документом и провайдером.

Из этого списка нам интересно только первые 3 компонента, а именно DataSource, ClientDataSet и DataSetProvider. Используя набор из этих компонент, можно обеспечить доступ к данным. Причем в случае, когда речь идет о BDE и таблицах Paradox, как правило, достаточно использовать лишь один из перечисленных компонентов - DataSource. Этот компонент имеет всего 4 собственных свойства - AutoEdit, DataSet, Enabled и State. Свойство Enabled похоже на свойство Active таблицы или Connected у базы данных, т.е. делает активным или неактивным соединение. А свойство AutoEdit, будучи включенным, обеспечивает возможность правки записей без написания какого-либо дополнительного кода. Свойство State информирует о том, в каком состоянии в текущий момент находится источник данных. Ну а самое важное свойство этого компонента - это, конечно же, DataSet, которое и определяет источник данных - таблицу, запрос и т.д.

Оставшиеся ClientDataSet и DataSetProvider могут понадобиться в том случае, если требуется обеспечить кэширование записей, например, для того, чтобы представить в виде таблицы источник данных типа dbExpress.

Рассмотрим пример, когда нам требуется обеспечить доступ к данным простой таблицы Paradox. Для этого нам на форме приложения понадобятся следующие компоненты: Database и Table из BDE, а так же DataSource из Data Access.

ПРИМЕЧАНИЕ

Хотя для таблицы (компонента Table) можно указать один из определенных в BDE псевдонимов без помощи компонента Database, по сложившейся традиции, а так же в целях удобства управления приложением, все-таки предпочтительнее использовать связку из Database и Table.

Теперь для свойства AliasName компонента Database выберем название имеющегося у нас псевдонима БД (DATA1), а в качестве значения свойства DatabaseName так же напишем DATA1. Таким образом, компонент Database будет видеть "настоящий" псевдоним DATA1, а все остальные компоненты приложения - псевдоним DATA1, определенный посредством компонента Database.

Если такое положение вещей вас смущает, то в качестве значения DatabaseName можно указать любое произвольное значение, например, MyDatabase - в таком случае у других компонентов, имеющих свойство DatabaseName среди возможных значений этого свойства, будет значиться и MyDatabase.

Тем не менее, мы остановимся на начальном варианте, и перейдем к компоненту Table, для которого нам так же придется установить значения для 2 свойств - DatabaseName и TableName. Для первого укажем DATA1, для второго - customer. Чтобы убедиться, что все сделано правильно, попробуем активировать связь с БД, для чего установим в истину свойство Active. Если все было сделано верно (включая создание псевдонима DATA1 и таблицы customer, о чем речь шла в предыдущей главе), то не только свойство Active таблицы изменится на истину, но и свойство Connected компонента Database так же изменится на истину.

Последнее, что осталось сделать - это поместить на форму компонент DataSource и установить значение его свойства DataSet в Table1. Таким образом, мы получим действующую связку из 3 компонентов, обеспечивающих все этапы взаимодействия с БД - от организации локального псевдонима и управления им (Database), до выбора конкретной таблицы с возможностью управления ее параметрами (Table) и предоставления ее данных любым другим компонентам (DataSource).

2 Таблица DB Grid

Теперь настало время рассмотреть собственно компоненты, которым могут понадобиться данные для представления. Все они расположены на закладке Data Controls. Прежде всего, это, конечно, специальная таблица для баз данных - DBGrid. Этот компонент является дальнейшим развитием обычной таблицы (StringGrid), но предназначен исключительно для отображения и редактирования связанной с БД информации. Соответственно, у DBGrid нет таких свойств, как Cells, Cols и Rows, поскольку все, что выводит этот компонент - есть прямое отражение текущего содержимого связанной с ним таблицы БД.

В то же время, у компонента DBGrid предусмотрен целый ряд специальных свойств, предназначенных для взаимодействия с БД. Прежде всего, это свойство DataSource, в котором указывают имя компонента-источника данных. Так, если на форму, где уже имеются настроенные соответствующим образом невизуальные компоненты Database, Table и DataSource поместить таблицу, в свойстве DataSource которой указать DataSource1, то мы сразу же увидим содержимое таблицы customer (рис. 1).

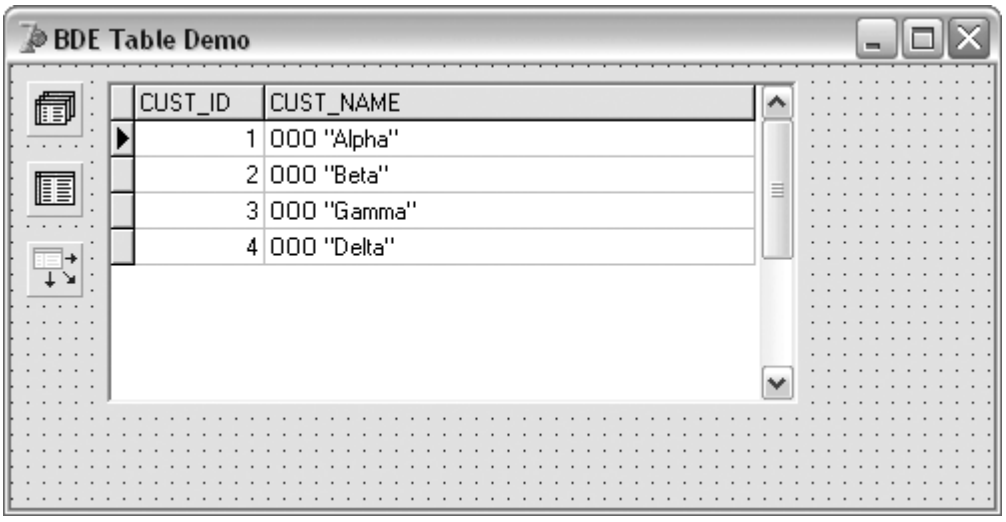

Рисунок 1 - Форма Delphi с таблицей БД

Следует сразу же отметить, что в качестве заголовков столбцов были использованы названия полей таблицы БД. Кроме того, можно увидеть, что столбец, содержащий числовые данные, имеет выравнивание по правому краю, а строковые - по левому. Таким образом, очевидно, что компонент DBGrid имеет более широкие возможности по оформлению таблиц, чем обычная таблица StringGrid. Возможно это благодаря другому свойству DBGrid - Columns, которое определяет оформление, количество и порядок следования столбцов с данными. Это свойство представляет собой коллекцию, состоящую из отдельных колонок таблицы. По умолчанию используется автоматический режим вывода, когда выводятся все поля данных с размерами, основанными на параметрах самих полей, заданных в выводимой таблице БД. Но поскольку во многих случаях выводить все поля не требуется, или же необходимо изменить параметры их вывода, как-то порядок следования, цвет, шрифт, или ширину поля, то все эти настройки доступны именно через свойство Columns. При этом каждый элемент этого свойства представляет собой объект типа TColumn, имеющий, в свою очередь, такие свойства, как заголовок (вместо стандартного названия поля в БД) выравнивание, цвет фона, шрифт, возможность правки и т.д. Доступ ко всем этим параметрам возможен через специальный редактор коллекций, который можно вызвать, дважды щелкнув по самой таблице в режиме разработки (рис. 2).

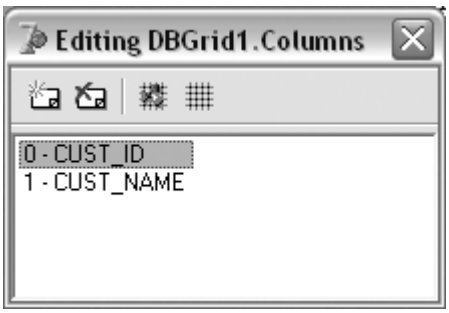

Рисунок 2 - Редактирование списка столбцов

На панели инструментов редактора столбцов имеется 4 кнопки - для добавления и удаления столбцов, а так же для автоматического заполнения и для сброса к начальным установкам. Выбирая поле из списка, и изменяя его

свойства в инспекторе объекта, вы тем самым изменяете параметры отображения соответствующего столбца. А меняя в редакторе строки местам (путем перетаскивания мышкой), вы меняете порядок вывода полей в самой таблице.

Таким образом, мы рассмотрели 2 наиболее важных свойства компонента DBGrid. Что касается всех собственных свойств, имеющихся у таблицы для баз данных, то они перечислены в таблице 1.

Таблица 1. Собственные свойства компонента DBGrid

Свойство	Тип	Описание
Columns	TDBGridColumns	Задает параметры вывода столбцов с данными
DataSource	TDataSource	Определяет источник данных для отображения в таблице
FieldCount	Integer	Указывает на число столбцов с данными, выводимых в таблице
Fields	array of TField	Предоставляет доступ к информации ячейки, находящейся в указанном столбце
Options	TDBGridOption	Определяет различные параметры отображения и поведения таблицы
ReadOnly	Boolean	Определяет, будет ли у пользователя возможность править данные в таблице
SelectedField	TField	Предоставляет доступ к информации в выделенной ячейке
SelectedIndex	Integer	Определяет номер текущего столбца
TitleFont	TFont	Определяет шрифт, используемый для вывода заголовков столбцов таблицы

Здесь следует отдельно выделить свойство Options, позволяющее настроить целый ряд различных параметров. Оно имеет следующие флаги:

- dgEditing - Делает возможной правку данных прямо в таблице. Этот флаг игнорируется, если включен флаг dgRowSelect;
- dgAlwaysShowEditor - Таблица будет постоянно находиться в режиме редактирования. В противном случае пользователь должен будет нажимать F2, Enter, или щелкать мышкой по полю, чтобы ввести новое значение;
- dgTitles - Делает видимыми заголовки столбцов;
- dgIndicator - Добавляет колонку, в которой будет отображаться индикатор выбранной записи;
- dgColumnResize - Делает возможным изменение размеров столбцов пользователем;
- dgColLines - Столбцы будут отделены разделительными линиями;

- dgRowLines - Записи будут отделены разделительными линиями;
- dgTabs - Делает возможной навигацию по ячейкам при помощи клавиш Tab и Shift+Tab;
- dgRowSelect - Записи будут выделяться целиком. При этом правка данных в таблице становится невозможной (т.е. флаги dgEditing и dgAlwaysShowEditor будут проигнорированы);
- dgAlwaysShowSelection - Выбранная ячейка будет выделена цветом даже если фокус вводе не находится на таблице;
- dgConfirmDelete - Будет выдаваться предупреждение, если пользователь захочет удалить запись в таблице (при помощи Ctrl+Delete);
- dgCancelOnExit - Предотвращает запись пустых записей;
- dgMultiSelect - Делает возможным выбирать несколько записей одновременно (с использованием клавиши Ctrl).

Например, если таблица должна будет использоваться лишь для навигации по БД и просмотра значений, то будет рациональным установить флаг dgRowSelect. С одной стороны, это автоматически отключит возможность правки и ввода данных непосредственно в самой таблице пользователем, а с другой - будет визуально выделять текущую запись, при этом достаточно наглядно показывая пользователю, что правка невозможна.

В типичном случае все производимые при разработке приложения настройки компонента DBGrid сводятся к тому, что, поместив его на форму, указывают связанный источник данных, после чего при помощи редактора определяют состав и вид столбцов данных. В случае при необходимости так же выставляют нужные значения для флагов в свойстве Options.

3 Навигация по таблице данных

Хотя в ряде случаев для обеспечения возможности навигации по таблице достаточно использовать лишь стандартные средства, которыми располагает компонент DBGrid, в ряде случаев бывает полезным предоставить пользователю более наглядный элемент управления для навигации по данным и для их правки. Более того, подобный компонент будет просто необходим, если для предоставления данных используется не таблица, а набор отдельных элементов, отображающих данные из одного конкретного поля.

Для этих целей предусмотрен специальный компонент - DBNavigator. С его помощью можно перемещаться по записям таблицы а так же выполнять операции типа вставки новой записи или подтверждения изменений. Внешне он представляет собой панель со следующими 10 кнопками:

- First - переход на первую запись в таблице;
- Prior - переход на предыдущую запись;
- Next - переход на следующую запись;
- Last - переход на последнюю запись;
- Insert - вставка новой записи перед текущей;
- Delete - удаление текущей записи с переходом на следующую;
- Edit - переводит источник данных в режим редактирования записи;
- Post - запись измененных данных из текущей записи в БД;
- Cancel - отмена изменений данных в текущей записи;

- Refresh - обновление данных в буфере источника.

Часть этих кнопок можно отключить, воспользовавшись свойством VisibleButtons. Еще одно свойство, влияющее на внешний вид компонента DBNavigator - это Flat. Установив его в истину, можно придать панели "плоский" вид. А при помощи свойства Hints можно задать пояснительный текст всплывающей подсказки для каждой кнопки.

Еще одно свойство - ConfirmDelete определяет поведение этого компонента: если для него установлено значение истины, то при попытке удаления записи (т.е. при нажатии на кнопку Delete) будет выдаваться соответствующее предупреждение.

Наконец, свойство DataSource, как и у всех других компонент представления данных БД, указывает на источник данных, связанных с данным компонентом.

Для примера добавим компонент навигации на форму с таблицей, подобной той, что изображена на рис. 1. Достаточно установить свойство DataSource компонента DBNavigator в то же значение, что и у одноименного свойства компонента DBGrid, чтобы получить связанно работающие компоненты. Например, при редактировании записей в таблице, состояние кнопок в навигационной панели будет изменяться в соответствии с возможными действиями (рис. 3).

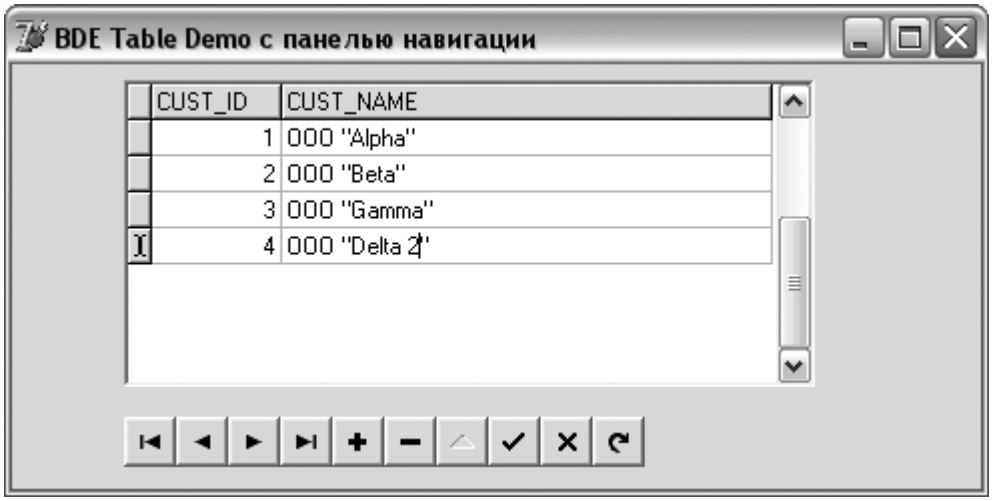

Рисунок 3 - Таблица DBGrid и панель навигации

Пример можно найти в каталоге Demo\Part4\SimpleTable.

4 Представление отдельных полей данных

Как мы уже знаем, для отображения таблиц БД в целом, используется табличный же компонент - DBGrid. В том же случае, когда надо отобразить содержимое лишь отдельных полей данных, используют соответствующие компоненты - DBEdit, DBImage, DBCheckBox и т.д., в зависимости от типа данных, которые требуется отобразить в том или ином случае.

Все эти компоненты являются специализированными вариантами обычных компонент, не связанных с БД:

- DBText - аналог текстовой подписи Label;
- DBEdit - аналог однострочного редактора Edit;

- DBMemo - аналог многострочного редактора (блокнота) Memo;
- DBImage - аналог компонента для вывода изображений Image;
- DBListBox - аналог списка ListBox;
- DBComboBox - аналог раскрывающегося списка ComboBox;
- DBCheckBox - аналог переключателя CheckBox;
- DBRadioGroup - аналог группы исключающих переключателей RadioGroup;
- DBRichEdit - аналог редактора форматированного текста RichEdit.

Основным отличием ориентированных на применение совместно с базами данных компонент является наличие у них свойства DataSource, при помощи которого они связываются с источником данных. Еще одно свойство - DataField, как раз и указывает на то поле, которое должно отображаться в данном компоненте. В остальном по своим функциональным способностям они повторяют свои не ориентированные на БД аналоги.

Что касается вариантов использования, то, например, мы можем заменить таблицу в предыдущем примере на пару компонент - DBText для отображения номера и DBEdit для вывода и редактирования названия. Соответственно, мы можем заменить табличное представление данных представлением типа "форма".

Для этого, удалив таблицу и разместив на форме компоненты DBText и DBEdit, установим для них обоих свойство DataSource в значение DataSource1, после чего для свойства DataField у метки выберем значение CUST_ID, а для этого же свойства у редактора - CUST_NAME. Не помешает также разместить на форме еще и 2 обычные метки (Label), при помощи которых можно вывести текст, поясняющий, что за информация выводится в том или ином поле (рис. 4).

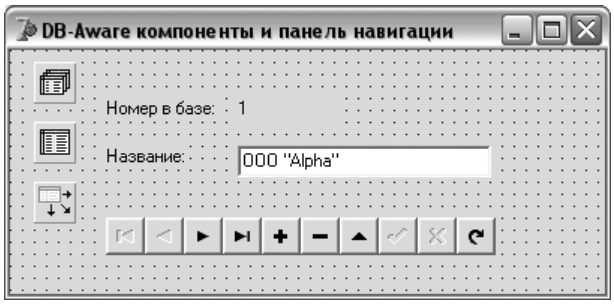

Рисунок 4 - Использование отличного от таблицы формата отображения данных

ПРИМЕЧАНИЕ

Здесь следует отметить тот факт, что данные могут автоматически преобразовываться в тот формат, который доступен компоненту. Например, целочисленное значение номера клиента было автоматически преобразовано в строку для вывода в DBText. Вместе с тем, возможности преобразований небезграничны, поэтому следует внимательно подбирать подходящие компоненты для отображения информации в каждом конкретном случае.

5 Компонент DBCtrlGrid

Особняком от других компонентов для отображения информации баз данных стоит компонент DBCtrlGrid, не имеющий прямых аналогов среди "обычных" компонентов VCL. Он состоит из набора однотипных панелей и позволяет отображать данные в произвольной форме. При этом каждая панель является платформой, на которой размещены простые БД-компоненты - такие, как DBText, DBEdit, DBCheckBox и т.п. У всех этих компонентов будет общий источник данных, который задается централизованно в свойствах самого DBCtrlGrid. Соответственно, останется лишь выбрать поля, которые помещенные на DBCtrlGrid компоненты будут отображать.

Во время разработки определяют единственную панель, по образу которой будут созданы ее точные копии. Например, можно взять за основу форму вывода таблицы клиентов в виде DBText и DBEdit, поместив на нее DBCtrlGrid, на котором, в свою очередь, расположить метку с редактором. В результате после запуска приложения эти 2 компонента будут продублированы на всех видимых панелях сетки (рис. 5).

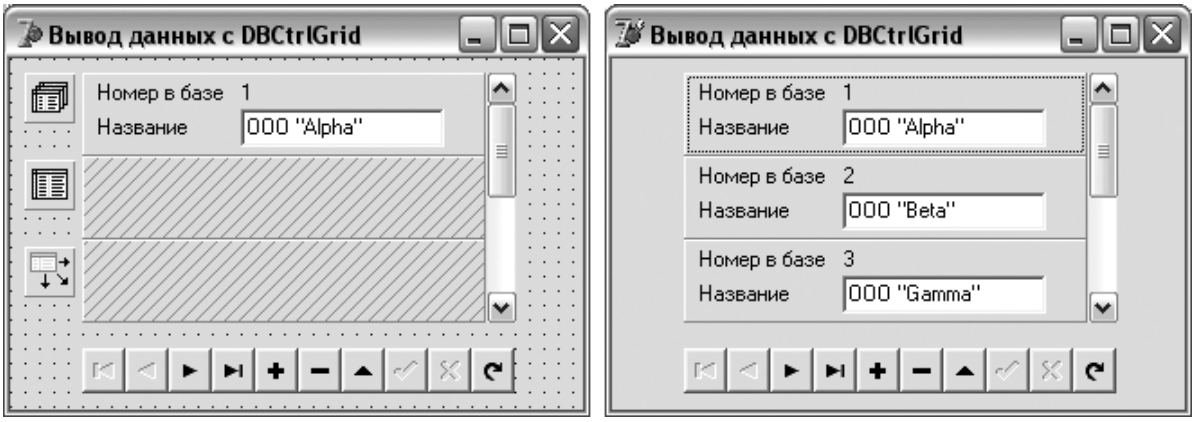

Рисунок 5 - Компонент DBCtrlGrid во время разработки (слева) и во время выполнения (справа)

Данные в таких панелях-ячейках могут располагаться как вертикально, так и горизонтально, либо вообще в качестве таблицы - в несколько рядов и колонок. Формат вывода панелей, как и их размеры, а так же ряд иных параметров этого компонента, определяется при помощи ряда свойств, перечисленных в таблице 2.

Таблица 2. Свойства компонента DBCtrlGrid

Свойство	**Тип**	**Описание**
AllowDelete	Boolean	Определяет, может ли пользователь удалить текущую запись, нажав Ctrl+Delete.
AllowInsert	Boolean	Определяет, может ли пользователь вставить новую запись, нажав клавишу

		Insert или добавить запись в конец, нажав Ctrl+Insert
ColCount	Integer	Определяет число столбцов с панелями
DataSource	TDataSource	Определяет источник данных для отображения
Orientation	TDBCtrlGridOrientation	Определяет порядок следования записей. Допустимые значения: goVertical (с вертикальной прокруткой) и goHorizontal (с горизонтальной прокруткой)
PanelBorder	TDBCtrlGridBorder	Определяет, должна ли быть рамка вокруг каждой панели. Допустимые значения: gbNone, gbRaised
PanelCount	Integer	Указывает на число видимых панелей
PanelHeight	Integer	Определяет высоту каждой панели в пикселях
PanelIndex	Integer	Определяет порядковый номе выбранной панели
PanelWidth	Integer	Определяет ширину каждой панели в пикселях
RowCount	Integer	Определяет число строк с панелями
SelectedColor	TColor	Определяет цвет фона активной панели
ShowFocus	Boolean	Определяет, должна ли отображаться дополнительная рамка вокруг панели при получении фокуса ввода

Размеры панелей определяются при помощи свойств PanelHeight и PanelWidth, а размеры компонента в целом определяются так же количеством панелей по горизонтали и по вертикали, определяемые через свойства ColCount и RowRount.

ПРИМЕЧАНИЕ

Не рекомендуется помещать на панели ресурсоемкие компоненты вроде блокнота или изображения, особенно если панелей много. Так же следует учитывать, что все компоненты будут иметь один и тот же источник данных, определенный для самого DBCtrlGrid.

Пример с использованием данного компонента совместно с панелью навигации можно найти в каталоге Demo\Part4\CtrlGrid.

6 Связывание данных в таблицах

Важной особенностью работы с БД является связывание данных в таблицах. И хотя такое связывание осуществляется невизуальным компонентом Table, наглядно продемонстрировать связывание можно лишь с использованием компонентов, отображающих данные таблиц, т.е. с использованием, например, рассмотренного в этой главе компонента DBGrid.

Очевидно, что для связывания данных между таблицами нам нужно иметь более одной таблицы. Поэтому для начала откроем Database Desktop и создадим новую таблицу, предназначенную для связывания с уже имеющейся у нас таблицей клиентов.

Пусть это будет таблица счетов (Bill) для этих самых клиентов. Таким образом, в новой таблице нам понадобится, как минимум, 3 поля: номер счета (BILL_ID), идентификатор клиента (BILL_CUST) и сумма (BILL_SUMM). При этом поле номера счета будет иметь автоинкрементный тип, и являться индексом, а поле с номером клиента - целочисленный тип и так же быть индексированным. Наконец, поле суммы будет денежного типа.

После создания структуры таблицы останется заполнить ее произвольными значениями - достаточно будет ввести по 1-2-3 записи для каждого из клиентов (рис. 6). Достаточно лишь помнить, что число, вводимое в поле идентификатора клиента должно соответствовать одному из значений в поле CUST_ID таблицы клиентов, т.к. если за этим не следить, то получатся "потерянные" данные, не связанные с другими, что является нарушением целостности БД.

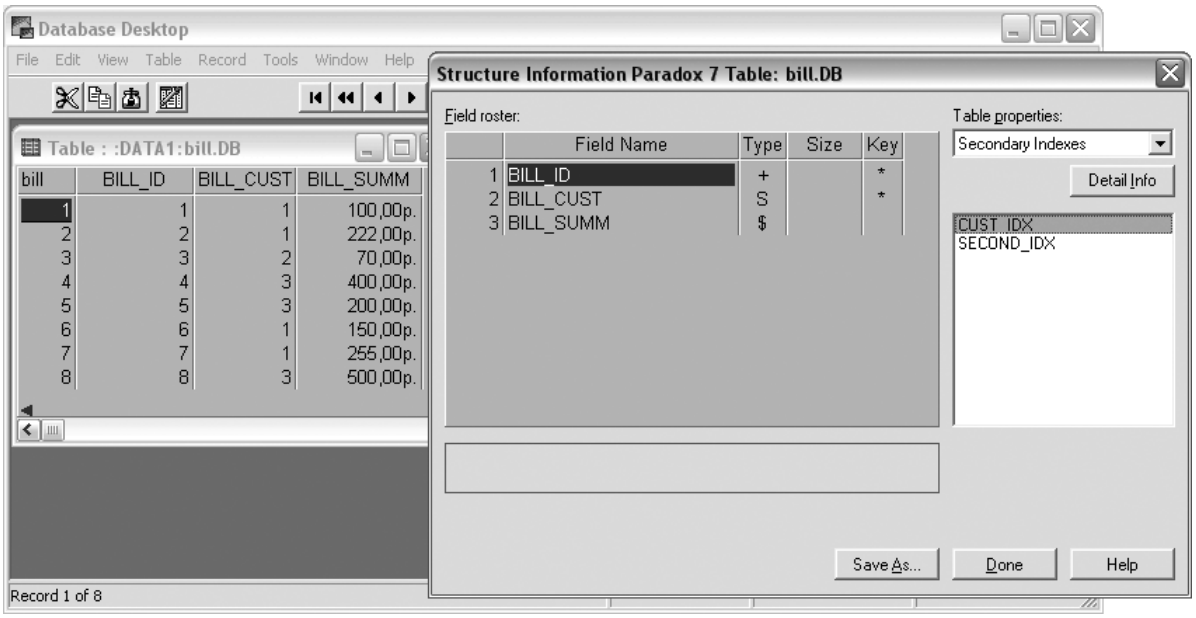

Рисунок 6 - Структура и содержимое таблицы счетов Bill

После подготовки исходных данных, перейдем к разработке приложения. Нам потребуется 1 компонент Database и по 2 компонента Table, DataSource и DBGrid. Для компонента Database установим те же значения, что и в предыдущем примере, т.е. AliasName и DatabaseName установим в DATA1, после чего свойство Connected можно установить в истину. Затем для первого "комплекта" из Table, DataSource и DBGrid так же устанавливаем значения, аналогичные предыдущему примеру, т.е. для Table1 установим свойство DatabaseName - в DATA1, а TableName - в customer.DB. Для DataSource1 установим свойство DataSet в Table1, и, наконец, для DBGrid1 установим свойство DataSource в DataSource1.

Затем подобную операцию следует провести для 2-го набора компонент, установив свойства подобным образом, за исключением того, что свойство TableName у компонента Table2 будет bill.DB, а во всех остальных случаях,

разумеется, будут использованы имена этого набора (т.е. Table2, DataSource2, и т.д.). Если теперь для компонентов Table1 и Table2 установить значение свойства Active в истину, то в DBGrid1 и DBGrid2 мы увидим содержимое таблиц Customer и Bill.

Теперь займемся собственно связыванием данных. Для этого нам надо в подчиненной таблице, которую здесь представляет компонент Table2, указать на главную таблицу и сослаться на связанное поле, по которому и будет производиться отбор. Поскольку главная таблица предоставлена своим источником данных (DataSource1), то его и следует указать в свойстве MasterSource компонента Table2. Затем для свойства MasterFields установим значение CUST_ID. Наконец, в свойстве INDEX_NAME укажем вторичный индекс таблицы Bill, т.е. тот, который индексирует поле BILL_CUST. В нашем случае он называется CUST_IDX.

Таким образом, остается запустить приложение, чтобы убедиться, что связи работают. Если все сделано правильно, то при выборе записи в таблице с клиентами, в таблице счетов будут отображаться только те данные, которые относятся к этому клиенту. Пример можно найти в каталоге Demo\Part4\OneToAny.

7 Компоненты синхронного просмотра

Специально для отображения связанной информации в БД, имеются 2 компонента, предназначенных именно для этих целей. Это компоненты DBLookupComboBox и DBLookupListBox. Оба они, хотя визуально и похожи на комбинированный и обычный списки, на самом деле, не являются потомками ни стандартных, ни БД-ориентированных компонентов, а происходят от общего для них класса TDBLookupControl, инкапсулирующего как список значений для просмотра, так и его механизм.

Соответственно, свойства этого класса наследуются обоими компонентами синхронного просмотра - как DBLookupComboBox, так и DBLookupListBox. Все они приведены в таблице 3.

Таблица 3. Общие свойства DBLookupComboBox и DBLookupListBox

Свойство	Тип	Описание
DataField	String	Определяет поле, которое будет отображаться данным компонентом
DataSource	TDataSource	Определяет источник данных для отображения в списке
Field	TField	Указывает на объект типа TField, который представляет собой данный компонент
KeyField	String	Определяет ключевое поле таблицы синхронного просмотра (ListSource), которое должно совпадать со

		значением поля, указанного в DataField
KeyValue	Variant	Представляет собой текущее значение ключевого поля
ListField	String	Определяет поле или список полей синхронного просмотра в таблице синхронного просмотра
ListFieldIndex	Integer	Определяет номер основного поля синхронного просмотра для случая, если в ListField указан список полей
ListSource	TDataSource	Определяет компонент источника данных (DataSource), связанный с таблицей синхронного просмотра
NullValueKey	TShortCut	Определяет сочетание горячих клавиш для сброса значения на неопределенное
ReadOnly	Boolean	Определяет, может ли пользователь изменять значения

Фактически, здесь следует запомнить лишь следующее: то, что отображается в самом компоненте синхронного просмотра (в обычном или в ниспадающем списке), задается парой значений для ListSource и ListField, а то, на основании чего происходит выборка текущего значения - в DataSource и DataField. При этом для связывания значений используется KeyField. Рассмотрим это на примере наших таблиц счетов и клиентов.

В предыдущем случае мы уже связывали таблицы средствами компонента Table. При этом первичным источником данных ("мастером") являлась таблица клиентов, выбирая запись в которой, мы могли видеть соответствующие ей значения в подчиненной таблице счетов. Но теперь рассмотрим такой вариант, когда первичной информацией являются сами счета. В таком случае, в общем-то, достаточно одной таблицы со счетами. Единственная проблема состоит в том, что в ней виден лишь номер клиента, что удобно машине, но не годится для человека. Соответственно, название клиента можно отображать, воспользовавшись как раз компонентом DBLookupComboBox.

Прежде всего, нам понадобится привычный уже набор из 5 компонент данных, а именно Database и 2 пары Table и DataSource. Первую пару (Table1 и DataSource1) в данном случае мы настроим на таблицу счетов, а вторую - на клиентов. При этом никаких значений для MasterSource и MasterFields нам в данном случае устанавливать не потребуется. После этого пометим на форму таблицу DBGrid и свяжем ее с таблицей счетов, установив свойство DataSource в значение DataSource1. Если база данных подключена, а таблицы активны, то в таблице сразу же будет отображена вся информация о счетах.

Теперь перейдем к собственно синхронному просмотру. Для этого поместим на форму компонент DBLookupComboBox и установим для его свойств следующие значения: DataSource - DataSource1, DataField - BILL_CUST, ListSource - DataSource2, ListField - CUST_NAME, и, наконец, свойство KeyFiled установим в CUST_ID.

Остается лишь запустить приложение и убедиться, что при выборе той или иной записи в таблице счетов, в комбинированном списке синхронного просмотра отображается название соответствующего клиента (рис. 7).

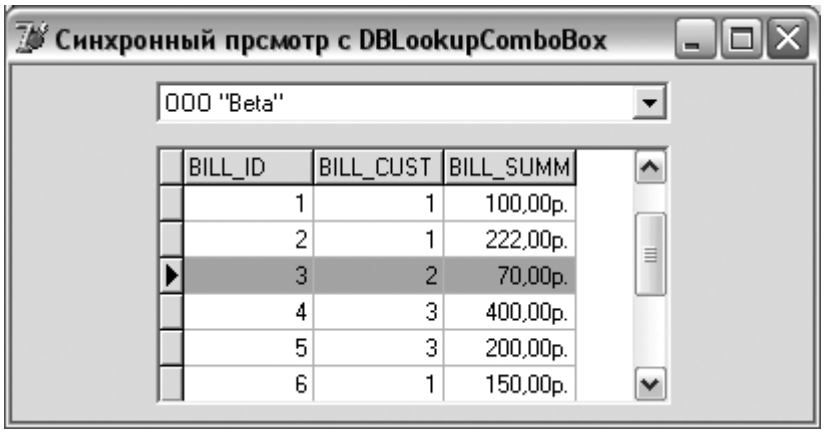

Рисунок 7 - Синхронный просмотр таблиц БД

Если же раскрыть сам список, то в нем можно будет увидеть весь список клиентов, а если выбрать из него другое значение, то цифра в колонке BILL_CUST так же поменяется. Таким образом, связь получается полноценная, в обе стороны.

Помимо рассмотренных свойств, отвечающих за организацию связи между данными, у компонента DBLookupComboBox имеются и собственные свойства, относящиеся к его визуальной части. Это DropDownAlign, DropDownRows и DropDownWidth, которые отвечают, соответственно, за выравнивание элементов в раскрывающемся списке, за их количество в нем и за ширину окна списка. Кроме них, для чтения во время выполнения программы доступны еще 2 свойства - ListVisible, указывающее на то, раскрыт ли список в данный момент, и Text, содержащее текущее значение списка в виде текстовой строки.

Что касается компонента DBLookupListBox, то с точки зрения организации синхронного просмотра он полностью аналогичен компоненту DBLookupComboBox. Различия касаются лишь визуальной формы представления, а так же собственных свойств, коих у синхронного списка всего 3 - BorderStyle, RowCount и SelecteItem. Первое отвечает за наличие обрамляющей рамки у списка, второе - указывает на количество видимых в списке рядов, а третье аналогично свойству Text у комбинированного списка, т.е. содержит выбранное значение в виде строки.

8 Модуль хранения компонентов данных

До настоящего момента мы рассматривали лишь простейшие случаи, с использованием 1-2 таблиц и такого же небольшого числа источников данных. Если же говорить о реальных приложениях БД, то число невизуальных компонентов, используемых в программе для доступа к данным, нередко исчисляется десятками. Такое их изобилие грозит превратить форму главного окна в одно сплошное нагромождение компонент, изрядно мешая работе. Кроме

того, одни и те же компоненты могут понадобиться в разных окнах приложения, в то время, как включать в список используемых модулей главное (или любое другое) окно только лишь для ссылки на данные не представляется идеальным вариантом. Поэтому в Delphi предусмотрено специальная оконная форма - DataModule, предназначенная исключительно для размещения на ней невизуальных компонент для доступа к данным.

Отличие окна DataModule от обычной формы состоит в том, что на нем можно размещать только невизуальные компоненты. Это могут быть не только компоненты для доступа к данным, но, в принципе, и любые другие, необходимые в разных частях приложения.

На таком окне можно совершенно свободно, без лишней скученности и без оглядок на пользовательский интерфейс, расположить весьма внушительное число необходимых компонентов (рис. 8).

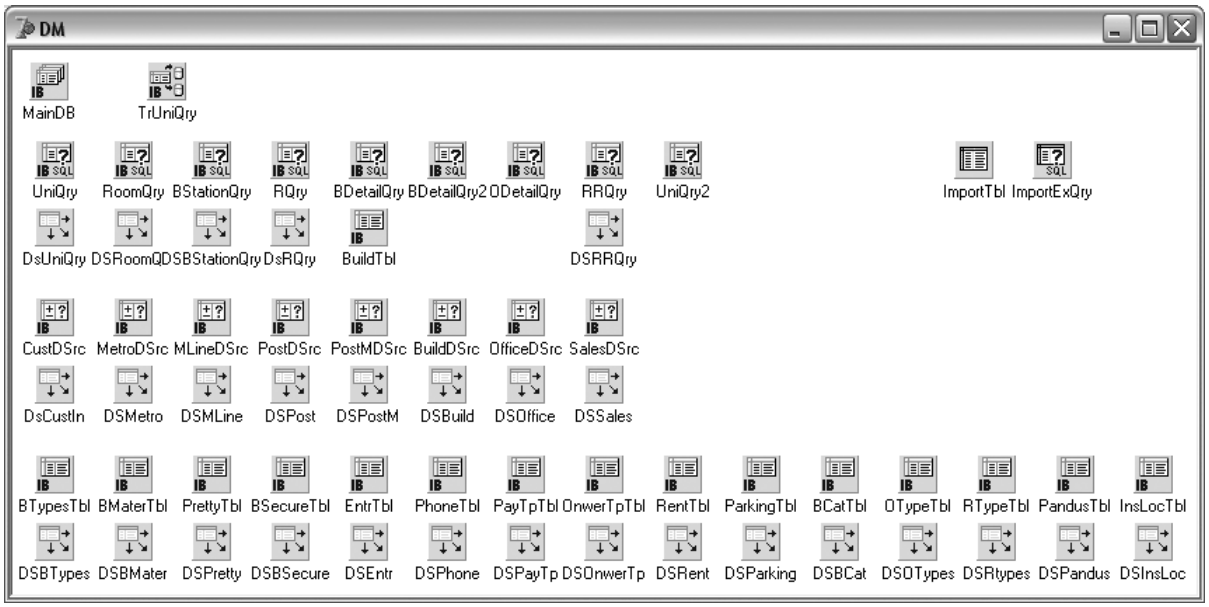

Рисунок 8 - DataModule может вместить в себя множество компонентов доступа к данным

Для создания окна DataModule следует из меню File > New выбрать пункт Data Module. После чего достаточно будет назначить имя этому модулю (например, DM) и сохранить файл, назвав его, скажем, data.pas. После этого, как и в случае с обычными формами, можно будет его включать в конструкцию uses. После этого становится возможным ссылаться на источники данных через стандартную точечную нотацию, используя имя модуля данных:

DBGrid1.DataSource:=DM.DataSource;

Пример, использующий вынесенные в отдельный модуль невизуальные компоненты, находится в каталоге Demo\Part4\AppDM.

Контрольные вопросы

1. Как создать таблицу в DatabaseDesktop?
2. Каким образом производится редактирование таблицы?
3. Как заполняются данные в таблицы?

4. Какие вы знаете базовые классы и компоненты для работы с СУБД в среде Delphi?

5. Что представляет собой модуль данных?

6. Для чего предназначены компоненты TTable, TDataSource, TDBGrid, TDBNavigator?

7. С помощью какого свойства источник данных связывается с таблицей?

8. Что представляют собой поля соответствия?

9. Какие существуют способы работы с таблицами?

10. Какой компонент в Delphi служит для создания запросов? На какой странице он находится?

11. Что представляет собой Построитель запросов?

12. Как в Построителе запросов можно просмотреть, какие существуют связи между таблицами?

13. Как указываются условия отбора записей?

14. Как сохранить запрос?

15. Как отобразить выполнение запроса на экране?

16. Что называется отчетом?

17. На какой вкладке палитры компонентов находятся компоненты для создания отчета?

18. Какой компонент является базовым при создании отчетов? Какие его свойства вы знаете?

19. Для чего предназначена полоса отчета? Какие свойства этого компонента необходимо установить?

20. Какие еще компоненты, предназначенные для работы с отчетами вы знаете?