**Практическая работа**

**Тема:** Базы данных

**Цель:** Разработка приложения баз данных

1. Краткие теоретические сведения

Визуальная разработка приложений баз данных. Все приложения СУБД, создаваемые в среде C++Builder, являются клиентами в архитектуре программного взаимодействия клиент/сервер. Клиент выдает запросы к серверу базы данных на получение или передачу информации. Сервер обрабатывает запросы от множества клиентов одновременно, координируя доступ к данным и их обновление.

Когда созданное с помощью C++Builder приложение в процессе работы обращается к базам данных, оно использует библиотеку . Эта библиотека устанавливается автоматически при установке C++Builder. Унифицированная технология BDE применяется во всех продуктах производства корпорации Borland: C++Builder,

Borland C++ и др.

При добавлении компонентов баз данных к форме приложения соединение с BDE происходит автоматически – никакого программирования не требуется. Во время выполнения программы BDE делает необходимые запросы и получает данные, заказанные свойствами каждого используемого компонента.

На рисунке 1 показано взаимодействие приложений с источниками данных с помощью BDE.

|  |
| --- |
| Приложения |

|  |
| --- |
| Инструменты формирования запросов |

|  |
| --- |
| Borland Database Engine (BDE) |

|  |
| --- |
| ODBC -драйверы |

|  |
| --- |
| SQL Links |

|  |
| --- |
| Серверные СУБД |

|  |
| --- |
| Локальные базы данных:  DBASE , Paradox и др. |

|  |
| --- |
| Локальный сервер InterBase |

|  |
| --- |
| TCP\IP, SPX/IPX и др. |

Рисунок 1.-. Связь приложений с источниками данных с помощью BDE

На рисунке 2 представлены компоненты доступа к базам данных.

Рисунок 2 - Компоненты доступа к базам данных

**Компоненты доступа к базам данных.** Невизуальные компоненты вкладки *Data Access* палитры компонент обеспечивают соединения с базами данных. Что позволяет разработчику сосредоточить внимание на обслуживании данных и не обращать внимания на организацию взаимодействия с пользователем. Компоненты осуществляют включение в программу элементы достуа к наборам данных –таблицам, запросам, хранимым процедурам и др.

Доступ к базам данных поддерживает его основа – 32-разрядный механизм BDE (Borland Database Engine).

Компонент *TDataSource* представляет собой интерфейс между прочими компонентами доступа к наборам данных и визуальными компонентами управления. Размещенными на форме. Именно посредством соединения с источником данных пользователь получает возможность отображения, навигации и редактирования содержимого баз данных. Компонент *TDataSource* также способен организовать парные связи таблиц между собой и поддерживать синхронизацию обмена. Свойство компонента *DataSet* указывает, с каким набором данных (таблицей, запросом) связан источник.

Компонент *TТable* представляет собой интерфейс между механизмом BDE и компонентом *TDataSource* который, в свою очередь, образует соединение с таким компонентом управления, как *TDBGrid.*

Именно посредством *TТable* обеспечивается доступ на этапе проектирования к живым данным из таблицы локальной базы данных: все записи или столбцы адресуемой таблицы сразу же становятся доступными для приложения.

Свойство *Active* устанавливает активное состояние связи с таблицей, чтобы можно было увидеть на форме живые данные. Свойство *DatabaseName* содержит псевдоним адресуемой базы данных или полный путь к ее каталогу. Свойство *TableName* – имя таблицы.

На рисунке 3 представлены визуальные компоненты вкладки *DataControls.*

Рисунок 3 - Визуальные компоненты вкладки *DataControls*

Визуальные компоненты вкладки *DataControls* палитры компонент обеспечивают взаимодействие пользователя с источниками данных приложения. Компоненты осуществляют включение в программу 12 элементов управления визуализацией редактированием записей или столбцов, хранимых в таблицах или запросах локальной базы данных.

Компонент *TDBGrid*  осуществляет отображение и редактирование записей, представляемых на регулярной сетке.

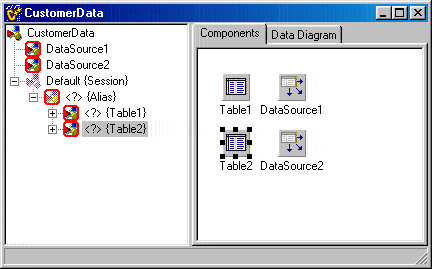
Пример: создание простой формы для демонстрационной базы данных BCDEMOS, в которой пользователь может прокручивать записи таблицы заказчиков CUSTOMER с отображением всех заказов, сделанных текущим заказчиком (таблица ORDERS), рисунок 4.

Рисунок 4 - Создание простой формы

1. Выполнить команду главного меню File -> NewDataModule, чтобы открыть контейнер нового модуля данных DataModule2. Из модуля удалить все компоненты и поместить две пары компонентов доступа к базам данных: TTable иTDataSource.
2. Установить свойства объекта ведущей таблицы Table1:

DatabaseName = BCDEMOS

TableName = CUSTOME.DB

# Name = CustomersTable

1. Установить свойства объекта ведомой таблицы Table2:

DatabaseName = BCDEMOS

TableName = ORDERS.DB

# Name = OrdersTable

1. Установить свойства объекта источника TDataSource1:

# DataSet = CustomersTable

# Name = CustomersSource

1. Установить свойства объекта источника TDataSource2:

# DataSet = OrdersTable

# Name = OrdersSource

1. Выполнить команду главного меню File -> New Form, рисунок 5.

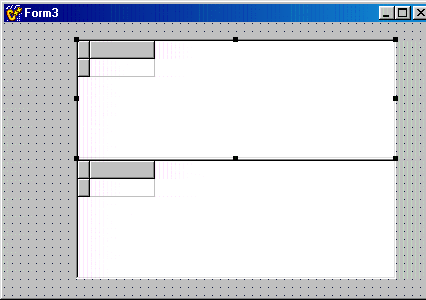


Рисунок 5 – Главная форма

1. Из вкладки DataControls нанести на форму две компоненты управления сеткой TDBGrid
2. Выполнить команду File ->Include Unit Hdr, чтобы указать, что данная форма должна использовать созданный модуль данных.
3. Установить свойство объекта первой сетки TDBGrid1:

# DataSource= CustomersData-> CustomersSource

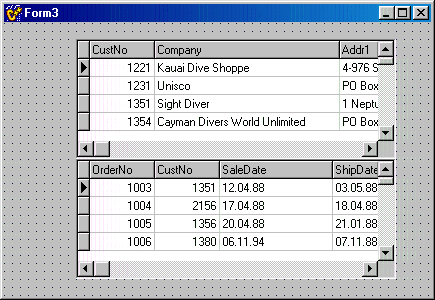
1. Установить свойство объекта второй сетки TDBGrid2:

# DataSource= OrdersData-> OrdersSource

1. Чтобы отобразить живые данные в сетках на форме, рисунок 6.

Установить свойств объекта Table1: Active ->true

Установить свойств объекта Table2: Active ->true

Рисунок 6 – Отображение данных таблиц

1. Открыть модуль данных и для таблицы OrdersTable установить свойство

# MasterSource= CustomersSource

чтобы связать ведущую таблицу CUSTOMER с ведомой таблицей ORDERS.

1. Дважды щелкнуть левой кнопкой мыши в графе значений свойствa MasterFields объекта OrdersTable, в открывшемся окне дизайнера связи полей Field Link Designer:

* выбрать номер заказчика CustNo (связующее поле таблиц) из выпадающего списка Available Indexes;
* задать CustNo в списках Detail Fields и Master Fields;
* нажать Add, чтобы добавить в список Joined Fields соединение CustNo-> CustNo;
* нажать ОК, чтобы подтвердить выбор, рисунок 7.

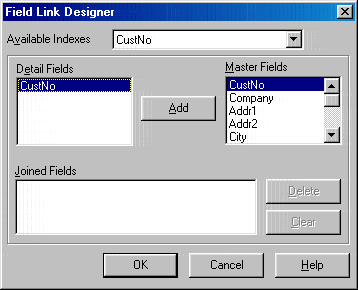


Рисунок 7 – Установка связей

1. Скомпилировать и выполнить приложение. Просмотреть работу связанных таблиц: при прокрутке записей таблицы заказчиков в таблице заказов видны только записи, относящиеся к текущему заказчику.

Попробуем создать простейшее приложение для редактирования таблицы БД. Для этого воспользуемся таблицей COUNTRY базы данных BCDEMOS, содержащейся в комплекте поставки C++ Builder.

Создадим новый проект и сохраним его главную форму как COUNT1.CPP, а сам проект как COUNT.MAK. Используя страницу Data Access палитры компонентов, разместим компоненты TTable и DataSource на форме. Установим свойство Database Name компонента Table1 равным BCDEMOS, свойство TableName равным COUNTRY.DB, а свойство Active равным true. Далее установим значение свойства DataSet компонента DataSource1 равным Table1.

Используя страницу Data Controls палитры компонентов, внесем компоненты DBGrid и DBNavigator в форму. Присвоим свойству DataSource обоих компонентов значение DataSource1, рисунок 8.

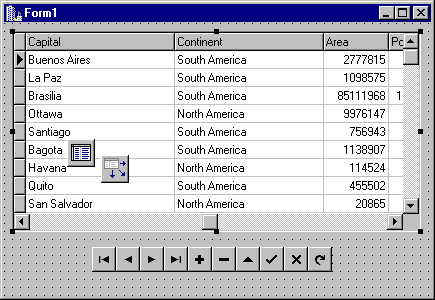


Рисунок 8 - Главная форма приложения COUNT

Полученный проект можно скомпилировать и протестировать.

Есть и более простой способ построить подобное приложение. Для этой цели можно воспользоваться утилитой SQL Explorer, вызываемой с помощью пункта меню Database/Explore. Выбрав страницу Databases в левой части окна SQL Explorer и щелкнув на значке "+" возле имени соответствующей базы данных, нужно выбрать опцию Tables. Затем в правой части окна нужно выбрать закладку Summary и перетащить с помощью мыши значок с именем нужной таблицы (в нашем случае COUNTRY.DB) на пустую форму. На форме появятся компоненты TDBGrid, TDataSource и TTable. После этого следует добавить DBNavigator, связать его с TDataSource и сделать таблицу активной.

Разработка приложения, использующего вычисляемое поле.

Таблица, используемая в предыдущем примере, содержит поля Area (площадь в кв.км ) и Population (население). Попробуем создать вычисляемое поле, содержащее п отность населения стран, представленных в этой таблице. Для этого запустим Fields Editor для компонента Table1, внесем все поля этой таблицы в набор данных и создадим новое поле с названием pop2. Выберем Float в списке Field type. Убедимся, что выбрана радиокнопка Calculated, и щелкнем на кнопке OK , рисунок 9.

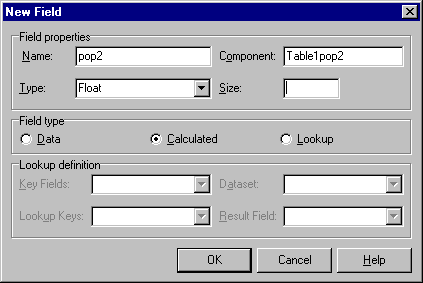


Рисунок 9 – Выбор поля

Создание вычисляемого поля. Создадим обработчик события OnCalcFields компонента TTable:

Table1pop2->Value=Table1Population->Value/Table1Area->Value;

Установим значение свойства DisplayFormat компонента Table1Pop2 равным .## . Запустим и протестируем приложение и убедимся в том, что вычисляемое поле действительно присутствует.

Компоненты TDBLookup. Компоненты DBLookup используются при наличии связанных таблиц, когда необходимо вывести на экран описательную информацию вместо поля, содержащего ее код.

C++ Builder предоставляет четыре компонента для просмотра и ввода значений в таблицы:

* Компонент TDBLookupListBox (страница Data Controls)
* Компонент TDBLookupComboBox(страница Data Controls)
* Компонент TDBLookupList (страница Win 3.1)
* Компонент TDBLookupCombo(страница Win 3.1)

Последние два компонента присутствуют в палитре компонентов с целью совместимости с формами, созданными в версии Delphi 1.0 (как известно, в C++ Builder можно использовать формы Delphi 1.0 и 2.0).

Компоненты DBLookupList и DBLookupListBox являются похожими на ListBox компонентами, созданными для просмотра значения в одной таблице, основанного на значении в другой таблице. DBLookupList и DBLookupListBox содержат конечный набор значений. Когда DBLookupList и DBLookupListBox используются для ввода данных, пользователь должен выбрать один вариант из списка. DBLookupList и DBLookupListBox позволяют вывести на экран набор вариантов, основанных на значении в другой таблице.

Компоненты DBLookupList и DBLookupListBox отличаются от компонента DBListBox тем, что позволяют согласовать выбранное значение из списка с текущей строкой другой таблицы БД, тогда как для DBListBox список значений для выбора определен заранее и не имеет отношения к таблицам БД.

Компоненты DBLookupCombo и DBLookupComboBox являются похожими на ComboBox компонентами, то есть они похожи на DBLookupList и DBLookupListBox, за исключением того, что пользователь может либо выбирать значение в списке, либо вводить новое значение. ComboBox, на который похожи DBLookupCombo и DBLookupComboBox, сочетает в себе возможности ListBox с возможностями компонента Edit. DBLookupCombo и DBLookupComboBox отличаются от компонента ComboBox тем, что позволяют согласовать выбранное значение с текущей строкой другой таблицы БД. При размещении компонентов DBLookupList, DBLookupListBox, DBLookupCombo или DBLookupComboBox на форме эта форма в приложении должна содержать DataSource и компонент - потомок TDataSet (например, TTable).

Разработка приложения, использующего компонент просмотра баз данных. В этом примере рассматривается связь таблиц CUSTOMER (компании-заказчики) и ORDERS (их заказы) из базы данных BCDEMOS и используется компонент DBLookupListBox для того, чтобы отобразить на экране имя компании, соответствующее значению CustID, находящемуся в таблице ORDERS.

Чтобы создать форму для выведения на экран выбранных из таблицы ORDERS полей, следует открыть новый проект и сохранить главную форму как DBLOOK1.CPP, а проект как DBLOOK.MAK. Далее нужно разместить на форме компоненты TTable, DataSource, DBGrid и DBNavigator. Далее следует присвоить свойству Database Name компонента Table1 значение BСDEMOS, свойству TableName - значение ORDERS.DB, а свойству Active - значение true. Свойству DataSet компонента DataSource1 присвоим значение Table1, а свойству AutoEdit - значение false. Свойству DataSource компонентов DBGrid1 и DBNavigator1 присвоим значение DataSource1. Далее используем Fields Editor, чтобы внести в набор данных Table1 поля OrderNo, CustNo, SaleDate, ItemsTotal, AmountPaid. Расположим поля в Fields Editor так, чтобы OrderNo был бы первым, а CustNo - вторым. Можно скомпилировать и сохранить это приложение, затем запустить и протестировать его.

Теперь внесем в приложение компонент DBLookupListBox и установим его свойство DataSource равным DataSource1, а свойство DataField равным CustNo. Поместим на форму новые компоненты TTable и DataSource. Установим свойство DatabaseName для компонента Table2 равным BCDEMOS, свойство TableName равным CUSTOMER.DB, а свойство Active равным true. Свяжем DataSource2 с Table2. Свяжем компонент DBLookupListBox1 с DataSource2, установив значение свойства ListSource компонента DBLookupListBox1 равным DataSource2, свойства KeytField равным CustNo, а свойства ListField равным Company. Можно скомпилировать и сохранить приложение, затем запустить и протестировать его.

В компоненте DBLookupListBox будет выделено название компании, соответствующее значению CustNo в текущей строке DBGrid, рисунок 10. Можно использовать DBLookupListBox для выбора заказчика по имени, при этом в таблице Orders будет устанавливаться соответствующее значение CustNo.

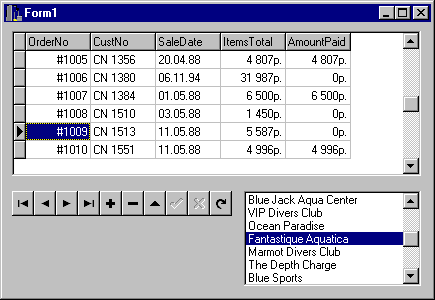


Рисунок 10 – Выбор записи

Приложение с компонентом DBLookupListBox.

**2. Задание для выполнения:**

1. Выполнить все приложения данных методических указаний
2. Скомпилировать проекты
3. **Контрольные вопросы:**

1. Какой механизм обеспечивает связь приложения с базой данных?
2. Какие визуальные компоненты базы данных вы знаете?
3. Какие невизуальные компоненты базы данных вы знаете?
4. Какие компоненты обеспечивают работу запросов в базах данных?
5. На каком языке записывается запрос?
6. Какие команды этого языка могут быть выполнены?