

## Лабораторная работа № 4

### Разработка приложений в системе C++Builder с использованием компонентов доступа к данным и отображения данных

**Цель работы:** научиться разрабатывать приложения с использованием компонентов, обеспечивающих доступ к данным и отображение данных, хранящихся в базе данных, управляемой системой MS SQL Server.

**Продолжительность работы** - 4 ч.

### Теоретические сведения

#### *Разработка приложений в системе C++Builder*

Запуск системы C++ Builder осуществляется на Терминале 4100 командой стартового меню Пуск | Все программы | Embarcadero RAD Studio 2010| C++Builder 2010. Для создания приложения используется команда главного меню системы C++Builder File | New | VCL Form Application - C++Builder.

В главном окне системы C++Builder (рис.1) располагаются несколько вспомогательных окон, используемых при разработке приложения, в частности:

- *окно дизайнера форм*, содержащее форму, в которой размещаются компоненты; изменение горизонтальных и вертикальных размеров формы осуществляется перетаскиванием ее границ с помощью мыши;
- *окно редактора* для ввода и редактирования текста программы;
- *окно инспектора объектов*, содержащее информацию о компоненте, выделенном в форме; информация содержится на закладках Properties (свойства) и Events (события). Свойства на закладке Properties можно располагать в алфавитном порядке названий (by Name)

или с группировкой по категориям (by Category) с помощью команды Arrange контекстного меню в окне инспектора объектов;

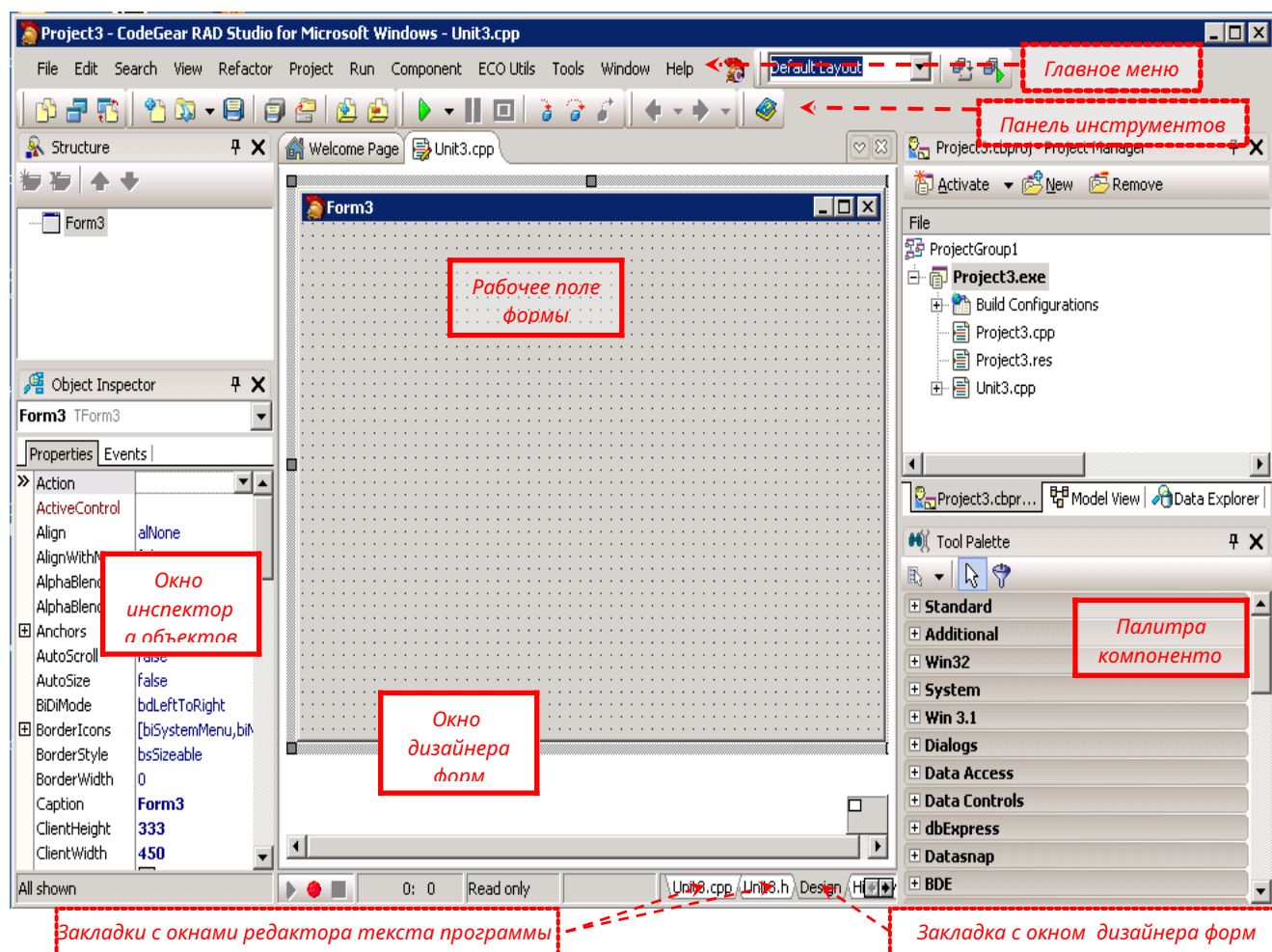



Рис.1. Главное окно системы

- *палитра компонентов*, состоящая из групп компонентов, каждый из которых имеет определенное функциональное назначение. В палитре компонентов в группах Data Access, BDE (Borland Database Engine – машина баз данных Borland) и dbGo собраны компоненты доступа к данным, а в группе Data Controls - компоненты отображения и управления данными. Для размещения какого-либо компонента в форме следует в палитре компонентов выбрать группу, содержащую нужный компонент, нажав на символ  рядом с названием группы, найти нужный компонент в появившемся списке, щелкнуть мышью по пиктограмме этого компонента и затем сделать щелчок мышью в том месте рабочего поля формы, где будет размещаться выбранный компонент.

Одновременно с размещением компонентов генерируется соответствующий код приложения на языке C++.

Созданное приложение запускается на выполнение командой главного меню Run | Run или нажатием кнопки с зеленым треугольником на панели инструментов.

### ***Компоненты, предназначенные для работы с базами данных***

Приложение для работы с БД создается в системе C++Builder посредством размещения в одной или нескольких экранных формах этого приложения специальных компонентов, которые настраиваются надлежащим образом, чтобы выполнять требуемые действия с данными.

В системе C++Builder компоненты для работы с БД реализуют наиболее употребимые функции и позволяют конструировать пользовательский интерфейс, подключать приложения к данным практически без написания программного кода.

Приложение, работающее с БД, обычно имеет в своем составе три вида компонентов (рис.2): компонент типа TTable или TQuery для связи с BDE и через него с БД либо компонент типа TADOTable или TADOQuery для подключения к БД с использованием технологии ADO (ActiveX Data Objects - объекты данных, построенные как объекты ActiveX); компонент типа TDataSource для соединения используемых приложением наборов данных с визуальными компонентами пользовательского интерфейса; визуальные компоненты для создания такого интерфейса. Количество компонентов и их взаимодействие определяются используемыми данными и решаемыми задачами.

Механизм взаимодействия компонентов с БД таков, что уже в процессе создания приложения можно просматривать данные без предварительной компиляции проекта.

Основным достоинством технологии ADO является ее естественная ориентация на создание “облегченного” клиента, для которого, в отличие от технологии BDE, не требуется устанавливать специальные программные средства на клиентской машине, поскольку необходимая поддержка обеспечивается операционной системой Windows.

При использовании технологии ADO на машине разработчика БД устанавливаются базовые объекты ADO и соответствующие компоненты C++Builder, обеспечивающие использование технологии ADO. На машине сервера данных устанавливается так называемый провайдер (поставщик) данных - надстройка над специальной технологией OLE DB, “понимающая” запросы объектов ADO и “умеющая” переводить эти запросы в нужные действия с данными. Взаимодействие компонентов ADO и провайдера осуществляется на основе универсальной для Windows технологии ActiveX.

На машине сервера создается и размещается база данных. В случае файл-серверных систем отдельные таблицы типа dBASE, FoxPro, Paradox и т.п. должны управляться соответствующим ODBC-драйвером, а в качестве провайдера используется Microsoft OLE DB Provider for ODBC drivers. Если по каким-либо причинам не найден нужный драйвер, файл-серверные таблицы можно преобразовать в формат СУБД Access. На их основе создается единый файл, содержащий все необходимые таблицы, индексы и прочие элементы БД. Такой файл управляется машиной баз данных Microsoft Jet 4.0 Database Engine, а провайдером служит Microsoft Jet 4.0 OLE DB Provider.

Если используется сервер данных Oracle или MS SQL Server, данные не нуждаются в каком-либо преобразовании, а в качестве провайдера применяется соответственно Microsoft OLE DB Provider for Oracle (или Oracle Provider for OLE DB) или Microsoft OLE DB Provider for SQL Server. Недостатком технологии ADO является то, что ею нельзя воспользоваться, если для соответствующей структуры данных (в частности, для БД многих популярных серверов - InterBase, Informix, DB2 и пр.) не создан нужный провайдер или ODBC-драйвер.

На машине клиента располагаются связанные компоненты TADOConnection и компоненты-наборы данных TADOTable, TADOQuery, TADOStoredProc, а также не показанные на рис.2,б

компоненты-наборы TADODataset и командные компоненты TADOCCommand. Каждый из этих компонентов может связываться с провайдером данных либо с помощью связанного компонента TADOConnection, либо, минуя его, через собственное свойство ConnectionString. Таким образом, компонент TADOConnection играет роль концентратора соединений с источником данных компонентов-наборов

Компонент TADOTable является аналогом BDE-компонента TTable и представляет в клиентской программе набор данных, состоящий из строк таблицы БД, имя которой содержит его свойство TableName.

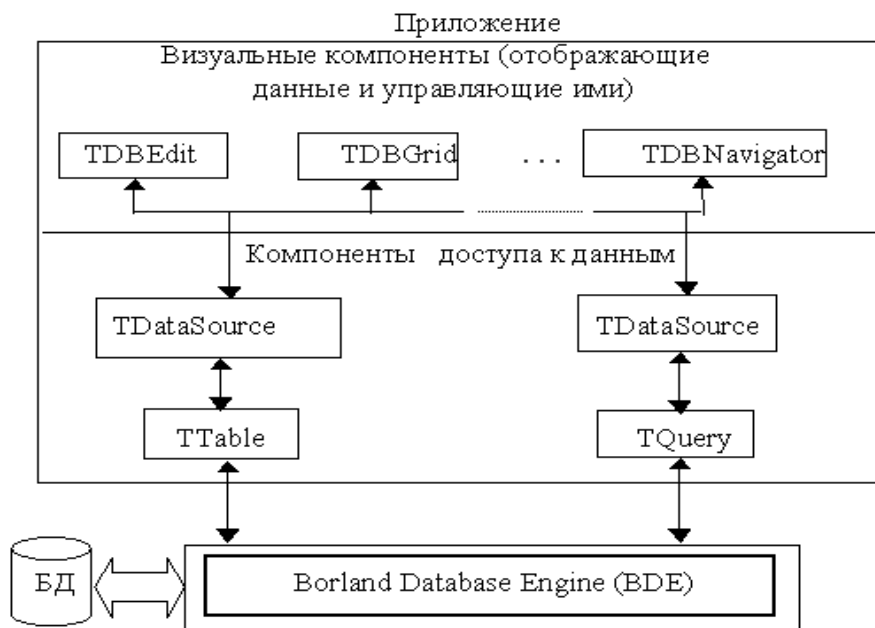
Компонент TADOQuery аналогичен BDE-компоненту TQuery и предназначен для получения строк из одной или нескольких таблиц БД и внесения изменений в одну из таблиц БД. Запрос на выполнение необходимой операции задается в свойстве SQL. Значение свойства SQL устанавливается либо в окне инспектора объектов, либо в программе методами Clear() и Add(). Для управления компонентом ADOQuery используются специальные методы Close(), Open(), ExecSQL().

Для связи с БД компоненты-наборы данных имеют два свойства: Connection и ConnectionString. В первое свойство помещается ссылка на специальный связной компонент TADOConnection, а во второе - строка связи. Эти свойства взаимоисключающие, т.е. установка значения одного из них автоматически сбрасывает значение другого. Строка связи содержит несколько параметров, разделенных точкой с запятой. В частности, предусмотрены параметры, обозначающие провайдера и местоположение БД.

Значение свойства ConnectionString устанавливается двумя способами:

- Use Data Link File - сослаться на специальный связной файл с расширением .udl, содержащий готовую строку связи;
- Use Connection String - сформировать строку связи самостоятельно.

При выборе второго способа указывается провайдер данных (например, Microsoft OLE DB Provider for SQL Server для СУБД MS SQL Server, как показано на рис.3) и после нажатия кнопки Next задаются местонахождение БД (например, имена сервера и БД на нем, как показано на рис.4) и параметры доступа к серверу (логин, пароль или другой способ аутентификации). Для проверки связи с БД можно воспользоваться кнопкой Test Connection.



*a*

Сервер

Клиент

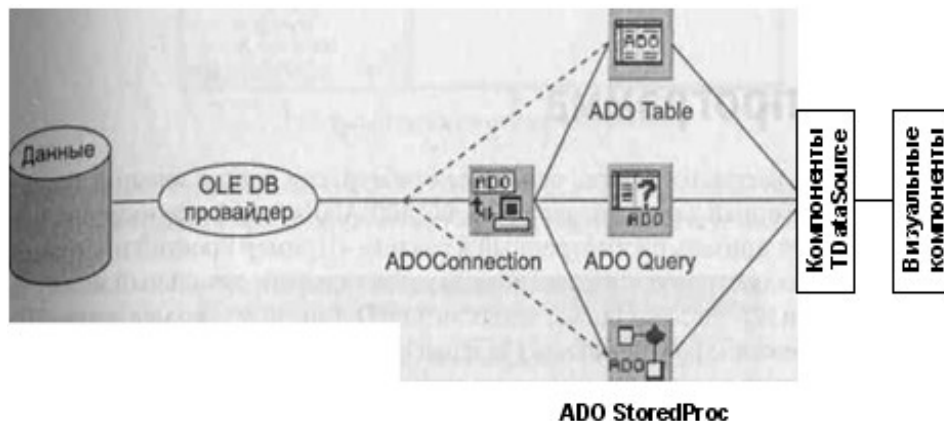


Рис.2. Компоненты, используемые для работы с базами данных по технологии BDE (а) и ADO (б)

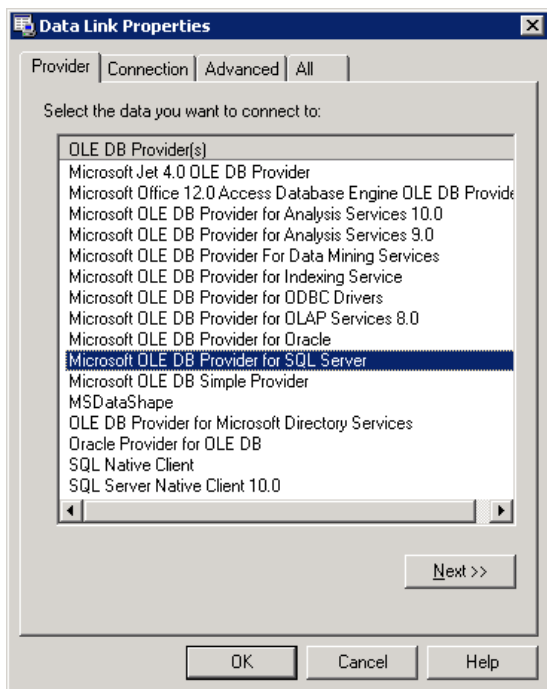


Рис.3. Окно выбора провайдера данных

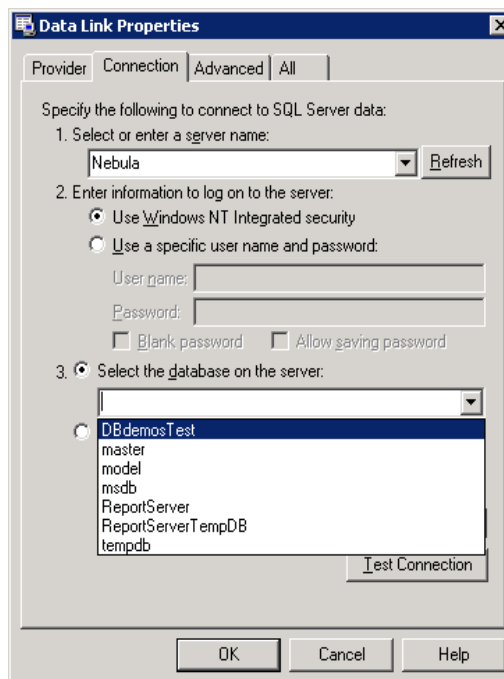


Рис.4. Окно выбора местонахождения БД и параметров доступа к БД

Данные становятся доступными программе после установки свойства `Active=True` у компонента `TADOTable` или `TADOQuery`.

В палитре компонентов группа `dbGo` содержит компоненты, обеспечивающие использование технологии ADO. Свойство `Active` компонента типа `TADOTable`, управляющее открытием таблицы, может устанавливаться вручную в окне инспектора объектов или программно методами `Open` (соответствует `Active=True`) и `Close` (соответствует `Active=False`). При установке свойства `Active=True` содержимое открытой таблицы появляется на экране в визуальном компоненте, отображающем данные. Следует помнить, что для компонента типа

TADOTable значения свойств `ConnectionString` и `TableName` доступны для изменения только при `Active=False`.

Компонент типа `TDataSource` находится в группе `Data Access`, а в группе `Data Controls` собраны компоненты для управления и отображения данных: `TBDNavigator`, `TDBGrid`, `TDBEdit`, `TDBText`, `TDBComboBox`, `TDBLookupListBox`, `TDBLookupComboBox`.

Компонент типа `TBDNavigator` (навигатор) управляет данными, отображаемыми в визуальных компонентах, использующих одноименный компонент типа `TDataSource`. Управление осуществляется с помощью кнопок навигатора (рис.5). Наличие конкретной кнопки задается в свойстве `VisibleButtons` установкой соответствующего идентификатора `nbXxx` в значение `True` (есть кнопка) или `False` (нет кнопки).

Компонент типа `TDBGrid` отображает содержимое таблицы в виде строк и столбцов, компонент типа `TDBEdit` - содержимое одного поля текущей строки таблицы, а компонент типа `TDBComboBox` отображает значение поля и задает список возможных значений этого поля. Отличие компонента типа `TDBText` от компонента типа `TDBEdit` заключается в недоступности отображаемого значения поля для изменения.

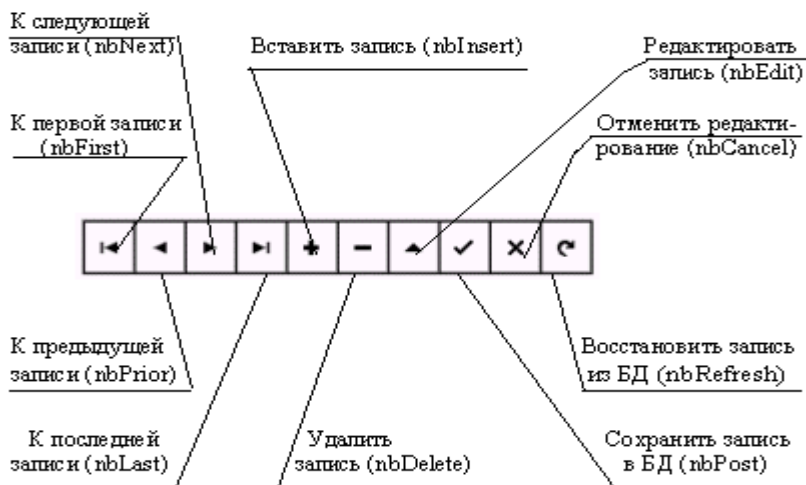


Рис.5. Кнопки навигатора



При создании приложения программисту после размещения необходимых компонентов в рабочем поле формы нужно установить связь между компонентами, выполнив следующую последовательность действий.

1. Задать спецификацию БД в свойстве *ConnectionString* компонента типа *TADOTable*.

2. Задать в свойстве *TableName* компонента типа *TADOTable* имя нужной таблицы. Имя таблицы выбирается из списка, который появляется при нажатии кнопки, расположенной в строке инспектора объектов, соответствующей свойству *TableName*.

3. В свойстве *Dataset* компонента типа *TDataSource* задать имя используемого компонента типа *TADOTable*. Значение свойства *Dataset* выбирается из списка аналогично тому, как выбиралось имя таблицы в п. 2.

4. Задать в свойстве *DataSource* визуального компонента типа *TDBGrid*, *TDBEdit*, *TDBText*, *TDBComboBox* или *TDBNavigator*, отображающего данные или управляющего данными, имя компонента типа *TDataSource*, через который осуществляется обмен и управление данными, хранящимися в таблице. Значение свойства *DataSource* выбирается из списка аналогично тому, как выбиралось имя таблицы в п.2. Для компонентов типа *TDBEdit* и *TDBText* следует задать в свойстве *DataField* имя поля таблицы, значение которого требуется отобразить, а для компонента типа *TDBComboBox* вместе со значением свойства *DataField* нужно в свойстве *Items* задать список возможных значений, присваиваемых полю текущей строки таблицы.

Взаимосвязь, существующая между таблицами в БД, обеспечивает корректность, или целостность, хранящихся данных. Например, демонстрационная БД (см. рис.16 в лабораторной работе № 1) содержит таблицу *ITEMS*, в которой значения поля *OrderNo* не должны отличаться от номеров заказов, хранящихся в таблице *ORDERS*. Устанавливать значение поля в строке таблицы (*ITEMS*) с учетом значений полей, хранящихся в другой таблице (*ORDERS*), позволяют компоненты типа *TDBLookupListBox* и *TDBLookupComboBox*.

С помощью этих компонентов связь с данными, хранящимися в таблице *ITEMS*, задается свойством *DataSource*; имя устанавливаемого поля *OrderNo* таблицы *ITEMS* является значением свойства *DataField*; связь с данными, взятыми из таблицы *ORDERS*, задается свойством *ListSource*; имя поля *OrderNo* таблицы *ORDERS*, значение которого выбирается для записи в одноименное поле таблицы *ITEMS*, является

значением свойства *KeyField*; имя поля *OrderNo* таблицы *ORDERS*, значение которого отображается на экране, является значением свойства *ListField*.

Компонент типа *TDBLookupListBox* или *TDBLookupComboBox* отыскивает в таблице, связанной с *ListSource*, строку, в которой значение поля с именем, указанным в *KeyField*, совпадает со значением поля с именем, указанным в *DataField*, и отображает из найденной строки значение поля, имя которого указано в *ListField*. Такие функциональные возможности позволяют при добавлении строки в таблицу *ITEMS* выбирать номер заказа из поля *OrderNo* таблицы *ORDERS* и записывать его в одноименное поле таблицы *ITEMS*.

### **Типы экранных форм для приложений, работающих с базами данных**

По внешнему виду пользовательского интерфейса экранные формы для приложений, работающих с БД, можно условно разделить на три типа: ввод/редактирование; сетка; главная/подчиненная. В этих экранных формах данные из таблицы БД отображаются в визуальных компонентах *TDBGrid* или *TDBEdit*.

В форме типа ввод/редактирование (рис.6) отдельные поля одной строки таблицы БД располагаются либо горизонтально, либо вертикально.

CustNo	Company	Addr1	
CN 1221	Kauai Dive Shoppe	4-976 Sugarloaf Hwy	
Addr2	City	State	
Suite 103	Kapaa Kauai	HI	
Zip	Country	Phone	FAX
94766-1234	US	808-555-0269	808-555-0278
TaxRate	Contact	LastInvoiceDate	
8.50%	Erica Norman	02.02.95 1:05:03	

Рис.6. Пример формы типа ввод/редактирование

Форма типа сетка (рис.7) отображает данные в стандартном табличном виде.

Форма типа главная/подчиненная (рис.8) позволяет просматривать одновременно содержимое нескольких связанных таблиц, одна из которых является главной (например, CUSTOMER), а другая подчиненной (например, ORDERS). При создании такой формы программисту необходимо выполнить следующую последовательность действий.

CustNo	Company	Addr1	Addr2
CN 1221	Kauai Dive Shoppe	4-976 Sugarloaf Hwy	
CN 1231	Unisco	PO Box Z-547	
CN 1351	Sight Diver	1 Neptune Lane	
CN 1354	Cayman Divers World Unlimited	PO Box 541	
CN 1356	Tom Sawyer Diving Centre	632-1 Third Frydenhoj	
CN 1380	Blue Jack Aqua Center	23-738 Paddington Lane	
CN 1384	VIP Divers Club	32 Main St.	
CN 1510	Ocean Paradise	PO Box 8745	
CN 1513	Fantastique Aquatica	Z32 999 #12A-77 A.A.	
CN 1551	Marmot Divers Club	872 Queen St.	
CN 1560	The Depth Charge	15243 Underwater Fwy.	
CN 1563	Blue Sports	203 12th Ave. Box 746	
CN 1624	Makai SCUBA Club	PO Box 8534	
CN 1645	Action Club	PO Box 5451-F	
CN 1651	Jamaica SCUBA Centre	PO Box 68	

Рис.7. Пример формы типа сетка

CustNo	Company	Addr1	Addr2
CN 1221	Kauai Dive Shoppe	4-976 Sugarloaf Hwy	
CN 1231	Unisco	PO Box Z-547	
CN 1351	Sight Diver	1 Neptune Lane	
CN 1354	Cayman Divers World Unlimited	PO Box 541	
CN 1356	Tom Sawyer Diving Centre	632-1 Third Frydenhoj	
CN 1380	Blue Jack Aqua Center	23-738 Paddington Lane	

OrderNo	CustNo	SaleDate	ShipDate	EmpNo	ShipTo
#1023	CN 1221	01.07.88	02.07.88	Emp# 0005	
#1076	CN 1221	16.12.94	26.04.89	Emp# 0009	
#1123	CN 1221	24.08.93	24.08.93	Emp# 0121	

Рис.8. Пример формы типа главная/подчиненная

1. Разместить в форме для каждой из связанных таблиц по одному комплекту из трех компонентов типа TADOTable, TDataSource, TDBGrid, обеспечивающих доступ, управление и отображение данных.

2. Задать свойства компонентов из комплекта для главной таблицы.
3. Задать свойства компонентов из комплекта для подчиненной таблицы, установив связь между полями связанных таблиц. Для связи подчиненной таблицы с главной предназначены свойства *MasterSource* и *MasterFields* компонента типа *TADOTable*, представляющего подчиненную таблицу. Задать значение свойства *MasterFields* можно с помощью конструктора связанных полей (Field Link Designer), который вызывается из инспектора объектов нажатием кнопки, расположенной в строке свойства *MasterFields*. В окне конструктора следует выбрать нужный индекс (индексный ключ) и установить связь между полями подчиненной (detail) и главной (master) таблиц.
4. Разместить навигатор и связать его с главной таблицей.

### Лабораторное задание

1. При домашней подготовке изучить описание лабораторной работы и законспектировать сведения о компонентах системы C++ Builder, работающих с БД, в виде письменных ответов на контрольные вопросы 1, 2, 5 - 12.
2. При домашней подготовке записать в отчет **SQL-операторы для запросов, содержащихся в ИЛМ, созданной для своего варианта задания в лабораторной работе № 2. Обязательно предусмотреть запросы, которые реализуются на языке SQL с использованием агрегатных функций, группировки и вложенных запросов.**
3. Выполнить на компьютере пп. 1 - 24, перечисленные в следующем разделе лабораторной работы, и показать результаты преподавателю.
4. Оформить отчет и защитить работу, ответив на заданные преподавателем вопросы.

### Порядок выполнения работы

1. Запустить систему C++ Builder 2010 на Терминале 4100.
2. Создать приложение с формой типа сетка для просмотра таблицы PARTS базы данных DBdemosTest (на сервере Nebula), содержащей информацию о поставках изделий, разместив в форме этого приложения все компоненты, необходимые для просмотра таблицы PARTS (рис.9), и установив в окне инспектора объектов свойства размещенных компонентов, указанные в табл.1.

3. Проверить работу приложения и сохранить приложение командой File | Save Project As в папке Lab4-Task1 на устройстве С терминального компьютера в папке Мои документы.

4. Создать приложение с формой типа ввод/редактирование для просмотра таблицы CUSTOMER, содержащей информацию о компаниях-покупателях (см. рис.6). В экранной форме для отображения значений полей текущей записи использовать компоненты типа TDBEdit, которые связаны с компонентом DataSource1 через свои

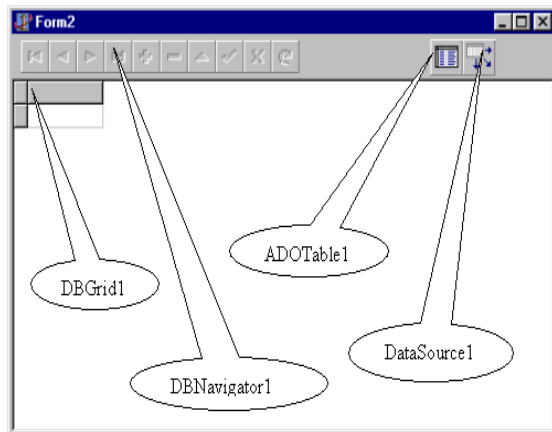


Рис.9. Форма для просмотра таблицы PARTS из БД DBdemosTest

свойства DataSource. Имя поля, значение которого отображается в компоненте типа TDBEdit, задать свойством DataField этого компонента. Надпись к полю задать свойством Caption компонента типа TLabel, принадлежащего группе Standard.

5. Проверить работу приложения и сохранить приложение в папке Lab4-Task2 на устройстве С терминального компьютера.

6. Создать приложение с формой типа главная/подчиненная для просмотра таблицы CUSTOMER и связанной с ней таблицы ORDERS, в которых содержится информация о компаниях-покупателях и сделанных заказах (см. рис.8). Информацию из каждой таблицы отобразить в компоненте типа TDBGrid.

7. Воспользоваться табл.2, в которой приведены значения свойств используемых компонентов, устанавливаемые при разработке приложения.

8. Проверить работу приложения и сохранить приложение в папке Lab4-Task3 на устройстве С терминального компьютера.

**Таблица 1**

**Свойства компонентов для однотобличного приложения**

Свойство	Значение	Примечание
ADOTable1 : TADOTable		
ConnectionString	Значение свойства ConnectionString устанавливается в инспекторе объектов щелчком на кнопке с многоточием в строке этого свойства. В активизированном окне отмечается режим Use Connection String (сформировать строку связи самостоятельно) и нажимается кнопка Build. В появившемся окне указывается поставщик данных Microsoft OLE DB Provider for SQL Server и после нажатия кнопки Next задаются местонахождение БД и параметры доступа к серверу (см. рис.3 - 4). Для проверки связи с БД можно воспользоваться кнопкой Test Connection. Формирование строки связи завершается нажатием кнопки ОК.	
TableName	parts	Имя файла с таблицей
Active	True	Управляет открытием таблицы
Name	ADOTable1	Имя компонента
DataSource1 : TDataSource		
DataSet	ADOTable1	Имя компонента, через который передаются данные из БД и в БД
Name	DataSource1	Имя компонента
DBGrid1 : TDBGrid		
DataSource	DataSource1	Имя компонента, через который осуществляется обмен и управление

Свойство	Значение	Примечание
		данными
Name	DBGrid1	Имя компонента
DBNavigator1 : TDBNavigator		
DataSource	DataSource1	Имя компонента, через который осуществляется обмен и управление данными
Name	DBNavigator1	Имя компонента
VisibleButtons	[nbFirst,nbPrior,...,nbRefresh]	Список используемых кнопок навигатора

**Таблица 2**

### Свойства компонентов для двухтабличного приложения

Свойство	Значение	Примечание
ADOTable1: TADOTable (главная таблица)		
ConnectionString	См. табл.1	
TableName	customer	Имя файла с таблицей
Active	True	Управляет открытием таблицы
DataSource1: TDataSource		
DataSet	ADOTable1	Имя компонента, через который передаются данные из БД и в БД
Name	DataSource1	Имя компонента
DBGrid1: TDBGrid		
DataSource	DataSource1	Имя компонента, через который осуществляется обмен и управление данными
Name	DBGrid1	Имя компонента
ADOTable2: TADOTable (подчиненная таблица)		
ConnectionString	См. табл.1	
MasterSource	DataSource1	Имя компонента типа TDataSource, связанного с

Свойство	Значение	Примечание
		главной таблицей
MasterFields	CustNo	Список полей главной таблицы для связи с подчиненной таблицей
IndexFieldNames	CustNo	Индексный ключ (список полей подчиненной таблицы, по значениям которых упорядочиваются записи)
Active	True	Управляет открытием таблицы
Свойство	Значение	Примечание
Name	ADOTable2	Имя компонента
TableName	orders	Имя файла с таблицей
DataSource2: TDataSource		
DataSet	ADOTable2	Имя компонента, через который передаются данные из БД и в БД
Name	DataSource2	Имя компонента
DBGrid2: TDBGrid		
DataSource	DataSource2	Имя компонента, через который осуществляется обмен и управление данными
Name	DBGrid2	Имя компонента
DBNavigator1: TDBNavigator		
DataSource	DataSource1	Имя компонента, чьим набором данных управляет навигатор
Name	DBNavigator1	Имя компонента

9. Приложение, созданное при выполнении п. 6, дополнить компонентами, обеспечивающими доступ, управление и отображение данных из таблицы ITEMS, и связать эту таблицу в качестве подчиненной с таблицей ORDERS.

10. Проверить работу приложения, которое должно отображать данные из трех связанных таблиц. Сохранить приложение в папке Lab4-Task4 на устройстве С терминального компьютера.



11. Приложение, созданное при выполнении п. 8, дополнить компонентами, обеспечивающими доступ, управление и отображение данных из таблицы PARTS, и связать эту таблицу в качестве подчиненной с таблицей ITEMS. Из таблицы PARTS требуется отображать только содержимое поля *Description* (описание поставляемого изделия) в компоненте типа TDBEdit.

12. Проверить работу приложения, которое должно отображать

Рис.10. Форма типа ввод/редактирование, дополненная компонентом TDBGrid для просмотра таблицы ITEMS

данные из четырех связанных таблиц, и сохранить приложение в папке Lab4-Task5 на устройстве С терминального компьютера.

13. Создать приложение с формой типа ввод/редактирование для просмотра таблицы ITEMS и затем дополнить созданное приложение компонентом типа TDBGrid, чтобы продублировать отображение содержимого таблицы ITEMS (рис.10).

14. Проверить работу приложения и сохранить приложение в папке Lab4-Task6 на устройстве С терминального компьютера.

15. Модифицировать приложение, созданное при выполнении п. 12, чтобы обеспечить целостность БД с учетом того, что в таблице ITEMS значения поля *OrderNo* не должны отличаться от номеров заказов, зафиксированных в таблице ORDERS; значения поля *PartNo* не должны отличаться от номеров поставок изделий, зафиксированных в таблице PARTS; значения поля *ItemNo* должны находиться в определенном диапазоне (например, 1 - 5).

Для обеспечения ввода только допустимых значений в поля *OrderNo* и *PartNo* таблицы ITEMS, а также установки номера поставки путем выбора названия изделия и запрета доступа к отображаемому значению поля *Discount*, следует в форме, созданной при выполнении п. 12, заменить компоненты типа TDBEdit компонентами других типов, указанными в табл.3 (рис.11).

**Таблица 3**

**Типы замещающих компонентов**

Отображаемое поле	Тип компонента	Отображаемое поле	Тип компонента
OrderNo	TDBLookupListBox	ItemNo	TDBComboBox
PartNo	TDBLookupComboBox	Discount	TDBText

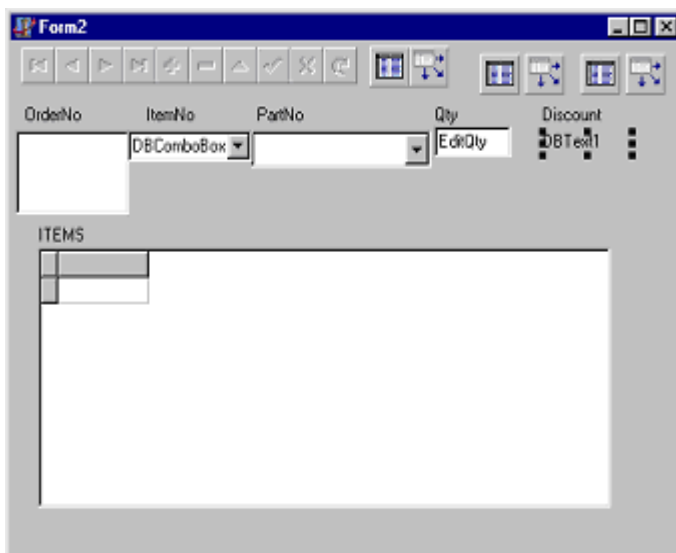


Рис.11. Форма с замененными и дополнительными компонентами

16. Настроить замещающие компоненты, чтобы:

- поле *OrderNo* (номер заказа) содержало значение, имеющееся в таблице ORDERS;
- поле *ItemNo* (порядковый номер изделия в заказе) содержало значение, выбираемое из диапазона 1 - 5;
- поле *PartNo* по номеру поставки находило в таблице PARTS и отображало в рабочем поле формы название соответствующего поставляемого изделия (товара);
- поле *Discount* (скидка) было доступно только для просмотра.

Для такой настройки таблицу (ORDERS или PARTS), связанную с таблицей ITEMS, представить в рабочем поле формы компонентами типа TADOTable и TDataSource, взаимосвязь таблиц задать установкой перечисленных в табл.4 свойств компонента типа TDBLookupListBox или TDBLookupComboBox, а свойства компонента типа TDBComboBox установить в соответствии с табл.5.

17. Проверить работу приложения, осуществив корректировку имеющихся и вставку новых строк в таблицу ITEMS, выбирая значения полей *OrderNo* и *PartNo* с помощью компонентов типа

TDBLookupListBox и TDBLookupComboBox; изменение данных наблюдать в дополнительно размещенном компоненте типа TDBGrid.

**Таблица 4**

### Свойства замещающих компонентов

Свойство	Значение	Примечание
DBLookupListBox1: TDBLookupListBox		
DataSource	DataSource1	Имя компонента, обеспечивающего доступ к таблице ITEMS
DataField	OrderNo	Имя поля в таблице ITEMS, значение которого отыскивается в таблице ORDERS
ListSource	DataSource2	Имя компонента, через который осуществляется обмен и управление данными из таблицы ORDERS, связанной с таблицей ITEMS
ListField	OrderNo	Имя поля в таблице ORDERS, значения которого отображаются в компоненте DBLookupListBox1
KeyField	OrderNo	Имя поля в таблице ORDERS, значение которого отыскивается по значению поля, указанного в свойстве DataField
DBLookupComboBox1: TDBLookupComboBox		
DataSource	DataSource1	Имя компонента, обеспечивающего доступ к таблице ITEMS
DataField	PartNo	Имя поля в таблице ITEMS, значение которого отыскивается в таблице PARTS
ListSource	DataSource3	Имя компонента, через который осуществляется обмен и управление данными из таблицы PARTS , связанной с таблицей

Свойство	Значение	Примечание
		ITEMS
ListField	Description	Имя поля в таблице PARTS, значение которого отображается в компоненте DBLookupComboBox1
KeyField	PartNo	Имя поля в таблице PARTS, значение которого отыскивается по значению поля, указанного в свойстве DataField

**Таблица 5**

**Свойства компонента DBComboBox1**

Свойство	Значение	Примечание
DBComboBox1: TDBComboBox		
DataField	ItemNo	Имя поля, значение которого отображается и устанавливается
DataSource	DataSource1	Имя компонента, обеспечивающего доступ к таблице ITEMS
Items	1 2 3 4 5	Список возможных значений, присваиваемых полю текущей записи; значения задаются в текстовом редакторе, который вызывается из инспектора объектов нажатием кнопки, расположенной в строке свойства Items

18. Разработать приложение для выполнения операторов языка SQL. Для этого:

- создать приложение и в форму этого приложения из группы Standard палитры компонентов поместить компонент Memo для ввода операторов SQL и три компонента Button для управления выполнением операторов SQL, из группы Data Access - компонент DataSource, из группы dbGo - компонент ADOQuery для взаимодействия с БД посредством технологии ADO, из группы Data Controls - компонент

DBGrid для отображения данных, извлеченных из БД (рис.12);

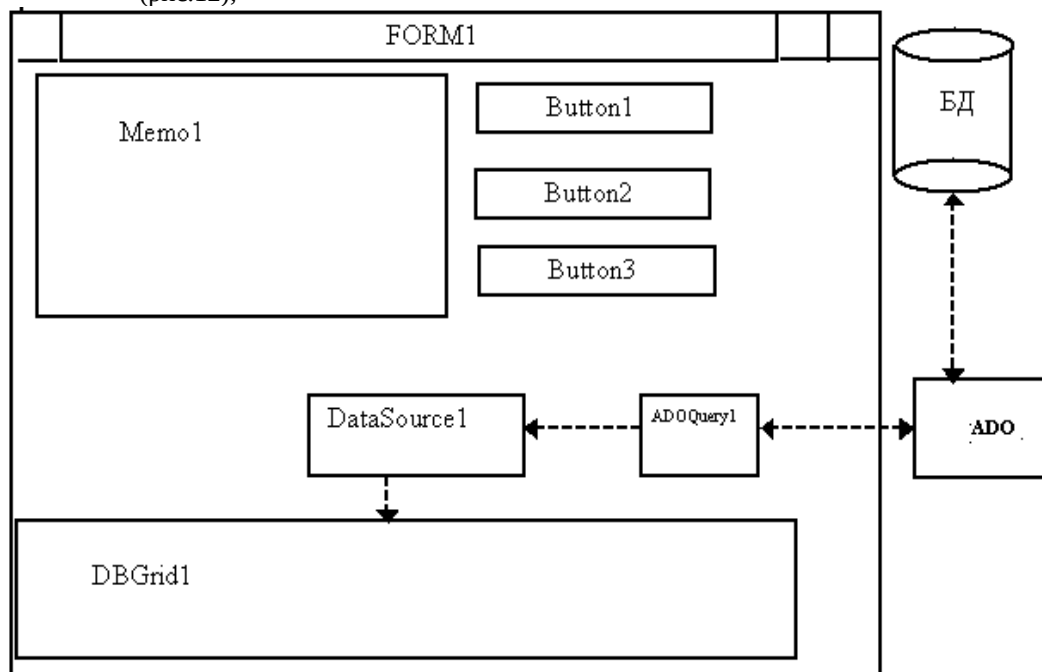


Рис.12. Расположение и связь компонентов

- настроить компоненты, размещенные в форме Form1 (рис.13), согласно табл.6;

Таблица 6

### Свойства компонентов для выполнения SQL-операторов

Компонент	Свойство	Значение
ADOQuery1	ConnectionString	См. табл.1
DataSource1	Dataset	ADOQuery1
DBGrid1	DataSource	DataSource1
Button1	Caption	SELECT
Button2	Caption	Update, Insert, Create, ...

Button3	Caption	Стереть результаты
Form1	Caption	Интерактивный SQL

- двойным щелчком на компоненте Button1 перейти в окно редактора и набрать следующие операторы:  
ADOQuery1->Close();  
ADOQuery1->SQL->Clear();  
ADOQuery1->SQL->Add(Memo1->Text);  
ADOQuery1->Open(); //выбрать данные из БД

Перейти в форму Form1, щелкнув на ее заголовке;

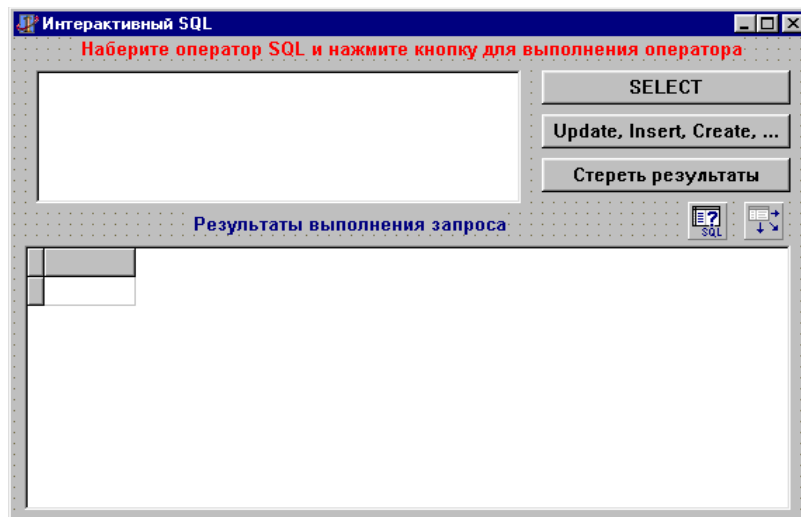


Рис.13. Форма с компонентами

- двойным щелчком на компоненте Button2 перейти в текстовый редактор и набрать следующие операторы:  
ADOQuery1->Close();  
ADOQuery1->SQL->Clear();  
ADOQuery1->SQL->Add(Memo1->Text);  
ADOQuery1->ExecSQL(); //изменить данные в БД

Перейти в форму Form1, щелкнув на ее заголовке;

- двойным щелчком на компоненте Button3 перейти в текстовый редактор и набрать следующие операторы:

```
ADOQuery1->Close();  
ADOQuery1->SQL->Clear();  
ADOQuery1->SQL->Add("select * from parts");  
// фиктивный оператор  
ADOQuery1->ExecSQL();
```

Перейти в форму Form1, щелкнув по закладке Design (см. рис.1).

19. Запустить созданное приложение на выполнение и с помощью операторов SELECT вывести содержимое таблиц CUSTOMER, ORDERS, ITEMS, PARTS.

20. Для указанных таблиц выполнить записанные в отчет по лабораторной работе № 1 операторы языка SQL с выборкой, сортировкой, группировкой, изменением и добавлением данных.

21. Сохранить приложение в папке Lab4-Task7 на устройстве С терминального компьютера.

22. Скорректировать приложение, чтобы иметь доступ к базе данных, спроектированной и созданной на сервере Nebula в лабораторной работе № 2.

23. Проверить работу скорректированного приложения, выполнив **включенные в отчет при домашней подготовке SQL-запросы для спроектированной базы данных.**

24. Сохранить приложение в папке Lab4-Task8 на устройстве С терминального компьютера.

### **Требования к отчету**

Отчет должен содержать:

- 1) название и цель работы;
- 2) схему взаимодействия клиентской программы и сервера посредством ADO и пояснения к схеме;
- 3) сведения о назначении компонентов и их свойствах.

### **Контрольные вопросы**

1. Какие компоненты системы С++ Builder предназначены для связи с БД?



2. Какие компоненты системы C++ Builder предназначены для отображения данных и управления ими?
3. Укажите достоинства и недостатки технологии ADO.
4. Приведите примеры форм типа ввод/редактирование, сетка, главная таблица/подчиненная таблица.
5. Какое свойство компонента типа TADOTable управляет открытием таблицы базы данных?
6. Какие значения может принимать свойство компонента типа TADOTable, которое управляет открытием таблицы базы данных, и как задавать эти значения вручную и программно?
7. Каким способом можно изменить состав кнопок навигатора?
8. Какие свойства компонента типа TADOTable задают связь этого компонента с конкретной таблицей базы данных?
9. Какие свойства компонентов типа TDataSource и TDBGrid (TDBEdit, TDBNavigator) и с какими значениями нужно задать, чтобы образовать связь компонентов, показанную на рис.2,а?
10. Какие свойства компонента типа TADOTable для подчиненной таблицы необходимо установить, чтобы задать связь с главной таблицей?
11. Укажите назначение свойств компонентов TDBLookupListBox, TDBLookupComboBox, TDBComboBox.
12. Укажите назначение свойств и методов компонента TADOQuery.