

Глава 16. Работа с DBF, Paradox и XML базами данных.....	388
16.1 Создание таблицы Paradox.	389
16.2 Русификация таблиц Paradox и DBF.	394
16.3 Моментальный поиск.	395
16.4 Создание псевдонимов.	397
16.5 Работа с XML таблицами.	399



Глава 16. Работа с DBF, Paradox и XML базами данных.

Я уже сказал, что очень люблю работать с Access базами данных, но моё пристрастие не распространяется на всех. Некоторые программисты любят старые таблицы Paradox и DBF, а некоторые уже перебазировались на новый и перспективный формат XML. Про эти форматы нельзя забывать, поэтому я не могу упустить их из виду.

Если ты думаешь, что в жизни хватит одного только Access, то сильно ошибаешься. Допустим ты пришёл на новую работу, где люди уже работают со старым форматом DBF. Из-за одного тебя никто не будет менять уже устоявшихся традиций и тебе придётся смириться с консерватизмом (ну и словечко), точнее сказать со старьём.

Конечно же, можно попытаться убедить начальство в том, что DBF и Paradox не надёжны и у них регулярно рушатся индексы. Можно убедить в том, что XML ещё не на столько гибок и отстаёт в возможностях, но на это нужно время и хотя бы какой-то промежуток времени тебе придётся работать со старыми форматами. А если не удастся убедить, то будешь работать так вечно, пока не поменяешь работу. Поверь мне, это проверенный вариант.

Так что не советую тебе расслабляться и переворачивать страницу, потому что эта глава достаточно важна. Тем более, что работа с другими форматами данных очень похожа на работу с ADO.



16.1 Создание таблицы Paradox.

Рaradox и DBF – это таблицы, а не базы данных. Если в одной базе Access могло храниться несколько таблиц, то у Paradox и DBF в одном файле храниться одна таблица. К тому же индексы хранятся отдельно от таблицы, что накладывает определённые неудобства.

Создание и работа с таблицами Paradox и DBF абсолютно одинаковы, поэтому я буду всё показывать на примере Paradox, потому что к нему отношусь с большим уважением.

Для создания первой базы данных тебе понадобится запустить отдельную программу *Database Desktop*, который входит в поставку с Delphi. После запуска выполни следующие действия:

1. Выбери меню File затем New и наконец Table
2. В появившемся окне, выбери из списка Paradox 7 .
3. Нажми "OK".

Перед тобой откроется окно как на рисунке 16.1.1. Можно сказать, что первая база данных готова. Теперь ты должен заполнить её поля, но сначала рассмотрим появившийся перед нами диалог.

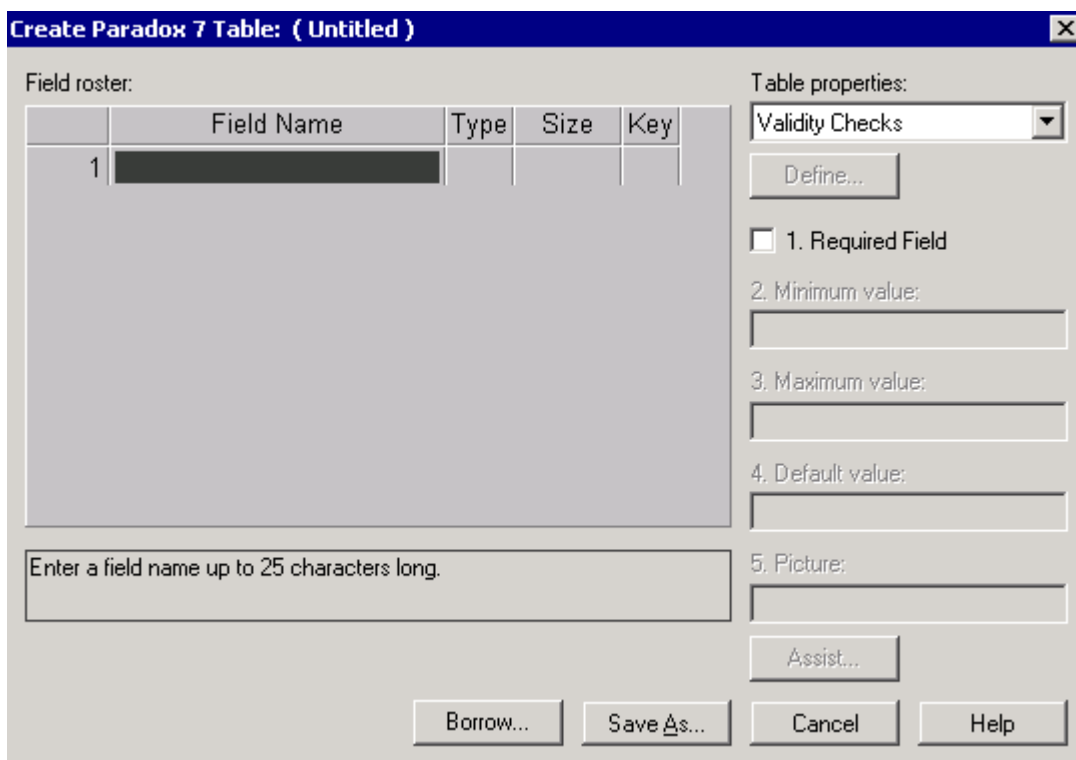


Рисунок 16.1.1. Окно редактирования полей таблицы Paradox

1. *Номер по порядку.* Database Desktop генерирует его автоматически и изменять ты его не можешь.

2. *Имя поля.* Здесь ты можешь называть свои поля как угодно, но только английскими буквами и нельзя использовать пробелы. Так что в отличии от Access, здесь могут быть проблемы с нормальным именованием полей.

3. *Тип поля.* Щёлкни в этой колонке правой кнопкой мыши и перед тобой появиться меню со всеми допустимыми типами, тебе необходимо только выбрать нужный.

4. *Размер*. Это размер поля. Не у всех типов полей можно менять размер, у большинства он задан жёстко. Размер в основном меняется у строковых типов (Alpha), бинарных (binary) и др.

5. *Ключ*. Если ты дважды щёлкнешь по этой колонке, то текущее поле станет ключевым, то есть по умолчанию по нему будет отсортирована вся таблица. Ключевыми могут быть только первые поля, то есть второе поле сможет быть ключевым только вместе с первым. Без ключевого поля невозможно добавлять новые записи в таблицу. Точно также, как и в Access создание ключа обязательно.

В качестве примера возьмём классический пример - приём заказов. Самое главное, это правильно представить себе и создать структуру базы данных. В нашем случае будет вестись учёт следующих полей:

1. ФИО Покупатель
2. Адрес
3. Дата заказа товара
4. Наименование заказанного товара
5. Количество заказанного товара

Теперь нужно продумать структуру. Если создать все эти поля в одной таблице, то будет не совсем эффективно. Если один и тот же покупатель возьмёт два товара, то у обеих строчек 1-е и 2-е поле будут содержать одинаковые данные. Будет лучше, если мы вынесем первые два поля в отдельную таблицу и потом будем использовать две связанные таблицы.

Итак, наша база данных будет состоять из двух таблиц. В первой будут следующие поля (после тире стоит тип поля, а в скобках размер):

Ключ 1 - autoincrement (ключевое)
ФИО Покупатель - alpha (размер 50)
Адрес - alpha (размер 50)
И соответственно 2-я:

Ключ 1 - autoincrement (ключевое)
Ключ 2 - Integer
Дата заказанного товара - date
Наименование заказанного товара - alpha (размер 20)
Количество заказанного товара - Integer

"Ключ 1" - это будет уникальное ключевое поле в обеих таблицах, поэтому поставь им значок ключевого. "Ключ 2" во второй таблице будет связан с "Ключ 1" из первой, пока его не делай ключевым, мы это сделаем чуть позже. Создай эти таблицы. Первую назови mast.db, а вторую child.db. Только помни, что имена полей должны писаться английскими буквами.

После того, как создашь вторую таблицу, ты должен сделать "Ключ 2" во второй таблице индексным, чтобы можно было связать две таблицы с помощью этого поля. Для этого нужно открыть таблицу child.db и из меню *Table* выбрать пункт *Restructure*. Перед тобой должно открыться окно, которое ты видел при создании полей таблицы (рисунок 16.1.2). теперь мы в этом же окне будем вносить изменения, а именно добавлять индекс.

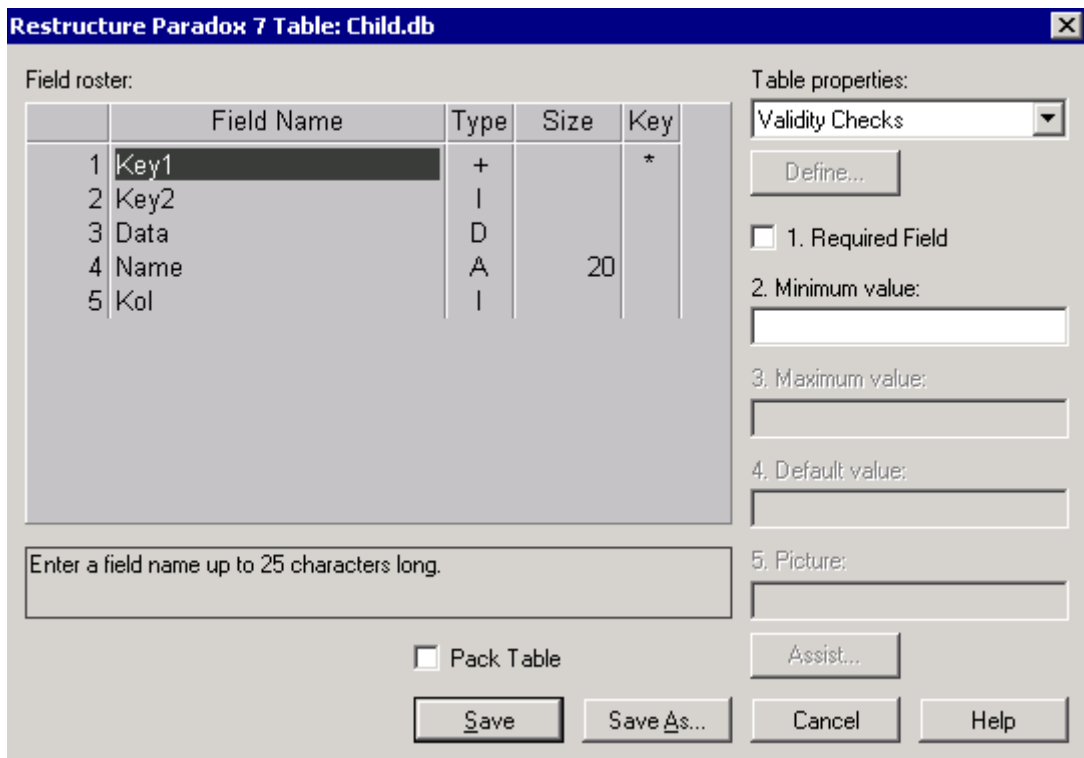


Рисунок 16.1.2. Окно редактирования полей таблицы Paradox

В выпадающем списке *Table properties* выбери *Secondary Indexes* (дополнительные индексы) и нажми кнопку *Define* (определить). Выбери свой второй ключ и перемести его в список *Indexed fields* (индексированные поля). Для этого надо нажать кнопку с изображённой стрелкой вправо (рисунок 16.1.3). Можешь нажимать "OK". У тебя запросят имя индекса, введи, например *hhh* и снова жми "OK". После этого сохраняй таблицу. Индексы готовы, теперь перейдём к последнему этапу подготовки базы.

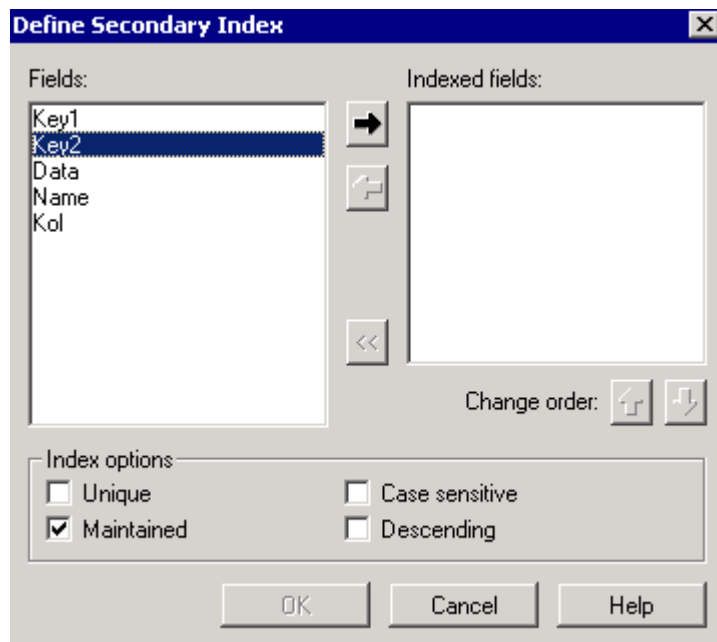


Рисунок 16.1.3. Окно редактирования полей таблицы Paradox

Теперь запусти SQL Explorer . Его главное окно ты можешь увидеть на рисунке 16.1.4. Здесь создаются псевдонимы (Alias) разным директориям с таблицами. Эти

псевдонимы сохраняются в реестре и потом программы при запуске могут по этим псевдонимам найти директорию таблицы и прочитать необходимые настройки, которые надо использовать при доступе к данным.

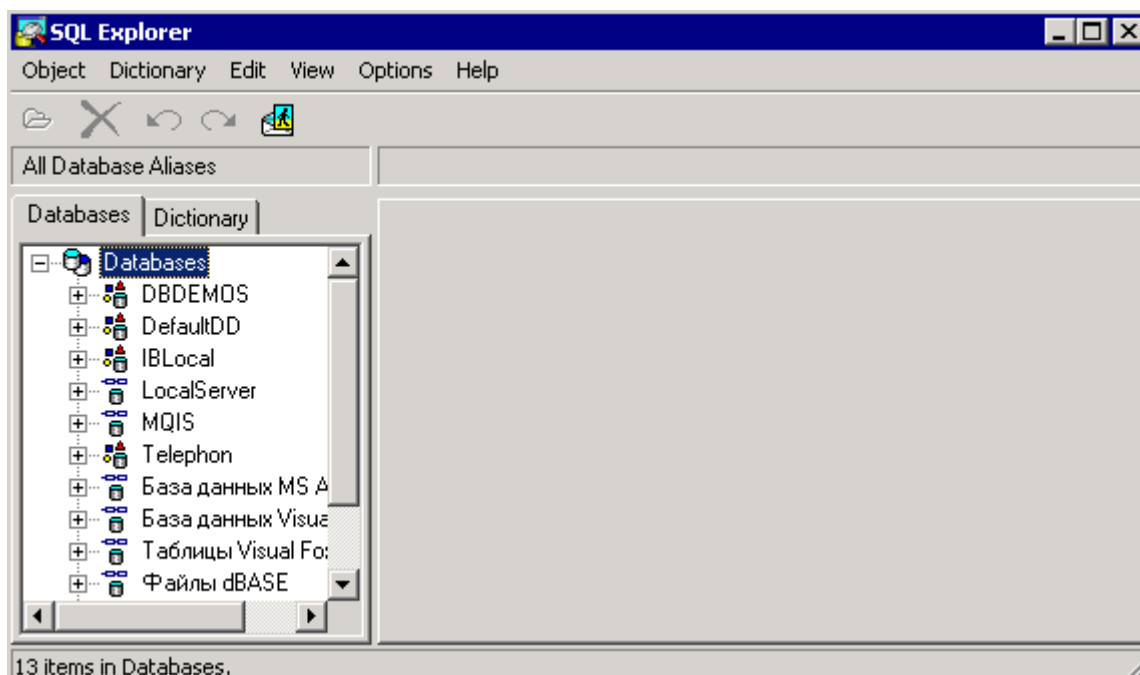


Рисунок 16.1.4. Окно SQL Explorer

Чтобы создать новое имя, нужно из меню *Object* выбрать пункт *New*. Перед тобой откроется окно, в котором *Database driver name* должен быть *STAMDDART*. Теперь жми "OK". Это создаст новый *Alias* (имя). Переименуй его в "Sales1".

Теперь в правой половине окна щёлкни по строке *PATH*. Перед тобой откроется окно выбора директории. Выбери ту, где находятся созданные нами таблицы и нажми "OK". Для сохранения того, что ты создал, выбери из меню *Object* пункт *Apply*.

Вот теперь таблицы готовы, имя создано и можно запускать Delphi, чтобы написать первую программу, которая будет работать с таблицами *Paradox*.

Создай новый проект. Помести на него из закладки *Data Access* два компонента *DataSource* и с закладки *BDE* два компонента *TTable* палитры компонентов. Для первого *DataSource* установи свойство *DataSet* в *Table1*, а у второго *Table2*. Теперь у *Table1* измени следующие свойства (желательно в такой последовательности):

DatabaseName выставь *Sales1* (это *Alias*, который мы создали в SQL Explorer).

TableName выставь в *mast.db*. Если ты всё правильно сделал, то имя этой базы будет в выпадающем списке этой строки.

Active выставь в *true*, чтобы активизировать таблицу.

То же самое сделай и с *Table2*, только в *TableName* выставь *child.db*. После того как ты всё это сделаешь, из палитры компонент *DataControls* поставь на форму два компонента *DBGrid*. В свойстве *DataSource* у одной из них выставь *DataSource1*, а у другой *DataSource2*. Постарайся расположить компоненты, как на рисунке 16.1.5.

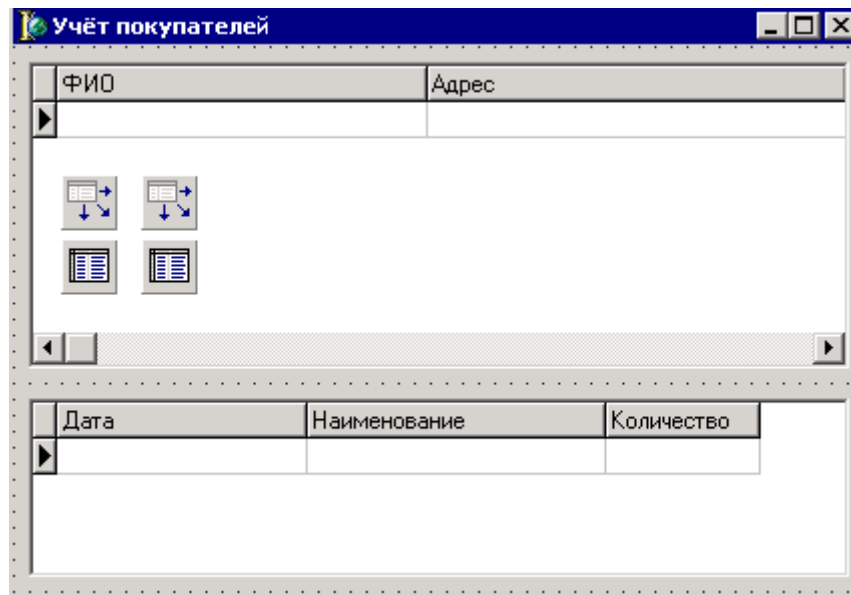


Рис 16.1.5. Расположения компонентов на форме

Уже можно запустить программу и убедиться, что всё работает. Но всё ещё не очень красиво, ведь таблицы ещё не связаны, и никому не нужно видеть ключи, да и подписи на английском. Выдели по *Table2*, здесь у тебя должна находиться *child.db*. В свойстве *MasterSource* выставь *DataSource1*. После этого дважды щёлкни в поле "MasterFields", перед тобой откроется окно как на рисунке 16.1.6. В верхнем списке *Available Indexes* выбери имя, которое ты задал, когда индексировал *Key2*, в данном случае это *hhh*. Теперь выдели *Key2* в левом окне и *Key1* в правом и нажми кнопку *Add*. Так ты добавил связь, которая будет использоваться между таблицами. Нажимай *OK* и снова ставь в свойстве этой таблицы *Active* в *true*, потому что при наведении связи таблица автоматически закрылась.

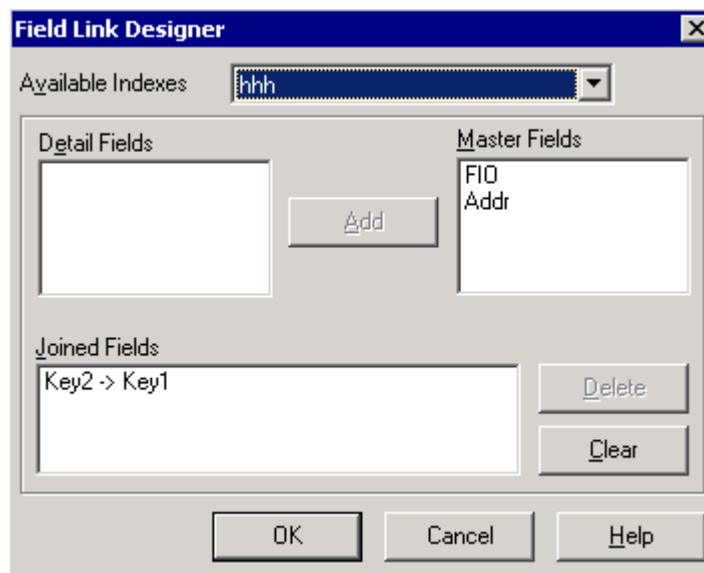


Рис 16.1.6. Окно связывания таблиц

Теперь дважды щёлкни по компоненту *Table2*. Перед тобой откроется окно (на рисунке 16.1.7) редактора свойств полей. С подобным окном мы работали, когда писали примеры с помощью ADO. В этом окне щёлкни правой кнопкой мыши, и в появившемся меню выбери пункт меню *Add All Fields*. После этого, все поля таблицы будут добавлены

в это окно. В свойствах *Key1* и *Key2* установи свойство *Visible* в *false*, чтобы спрятать индексы. В свойстве поля *Data* установи *DisplayFormat* в *dddddd*, а *EditMask* в *99/99/9999*. У всех полей свойство *DisplayLabel* отвечает за имя отображающее в компонентах это поле, поэтому напиши здесь у всех нормальные русские имена.

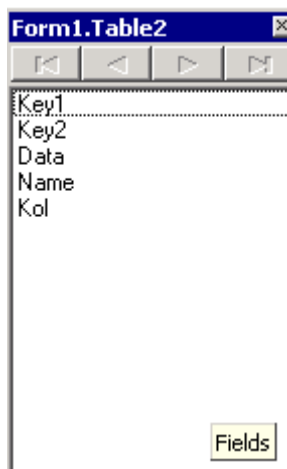



Рис 16.1.7. Окно свойств полей

После того, как закончишь с этой таблицей, сделай подобные операции со второй. Нужно будет так же спрятать ключевые поля и написать нормальные подписи к полям.

Твое первое приложение работающее с базой данных Paradox готово. Как видишь, работа с такими базами практически не отличается от работы с ADO. Единственное отличие – используются другие компоненты, а именно закладка BDE. Но большинство свойств компонента *TTable* похожи на свойства компонента *TADOTable*. У них есть все необходимые свойства, такие как *First*, *Next*, *Prev*, *Last*, *Edit*, *Paste* и многие другие.

Для создания запросов к таблицам Paradox и DBF нужно использовать компонент *Query* с закладки BDE. У него так же есть свойство *SQL*, в котором надо писать SQL запрос. Но так как таблицы Paradox и DBF – это таблицы, а не базы данных, то не надо указывать никаких строк подключения. Достаточно только ввести запрос (имя таблицы ты укажешь в запросе в поле *from*) и выполнить его сделав свойство *Active* равным *true*.

С таблицами Paradox можно так же создавать поисковые и вычисляемые поля и здесь особых отличий от ADO нет. Всё это связано с тем, что в обоих компонентах использована одна и та же основа.

 На компакт диске, в директории \Примеры\Глава 16\Paradox ты можешь увидеть пример этой программы.

В примере я не указывал у таблиц псевдонимы (Alias). Это не обязательно, но просто так принято для удобства. Если не указывать у таблиц в свойстве *DatabaseName* никакого псевдонима, то таблица будет искаться в текущей директории, где и запускной файл. Можно ещё указывать полный путь в свойстве *TableName*.

16.2 Русификация таблиц Paradox и DBF.

Для работы с таблицами Paradox и DBF мы используем компоненты с закладки BDE. Это не просто название закладки – это целый набор драйверов, программная надстройка, через которую происходит работа с таблицами. Эта

надстройка устанавливается вместе с Delphi или её можно найти на диске отдельным установочным файлом.

По умолчанию BDE работает с таблицами в кодировке не поддерживающей русский язык. Для русификации нужно запустить программу BDE Administrator. Её главное окно похоже на SQL Administrator. Перейди в этом окне на закладку Configuration (рисунок 16.2.1) и открой в дереве ветку *Configuration – Drivers – Native*. Здесь выбери пункт Paradox и в правой половине окна ты увидишь настройки доступа к таблицам Paradox.

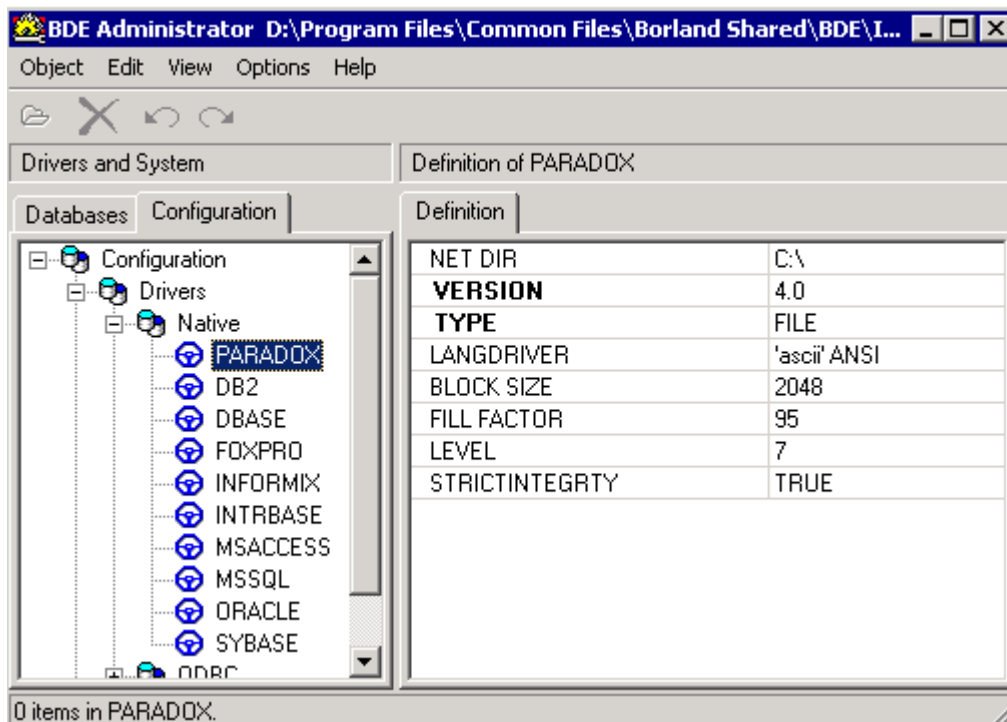


Рис 16.3.1. Обновлённая форма программы

Здесь нужно изменить параметр *LANGDRIVER* – драйвер языка. По умолчанию у меня стоит *ascii*, при котором русские буквы превращаются в непонятно что. Выбери у этого параметра в выпадающем списке *Pdox ANSI Cyrillic*. Теперь щёлкни в окне слева (в дереве настроек) по пункту Paradox и выбери в появившемся меню пункт Apply, чтобы сохранить настройки. После этого появиться окно с подтверждением о сохранении данных и после этого предупреждение о том, что для получения эффекта нужно перезапустить все программы работающие с BDE.

Теперь выбери в дереве пункт *DBASE* и у него в настройках выбери драйвер языка *dBASE RUS cp866*. Сохрани эти настройки.

Теперь твои таблицы будут правильно отображать русские буквы и ты сможешь работать с ними на родном и понятном нам языке.

16.3 Моментальный поиск.

Поиск одного поля с помощью SQL запроса или фильтра, просто ужасно глупая затея. Этот поиск будет проходить достаточно долго долго. Но есть способ лучше - Рондо. Точнее сказать поиск по ключевым полям. Этот поиск происходит практически моментально, даже на больших базах данных. Недостатки - отсутствие шаблонов и не возможно использовать привязанную таблицу.

Моментальный поиск мы не рассматривали в главе про ADO, потому что это особенность Paradox и DBF таблиц. Так что я сделаю это сейчас.

За счёт чего достигается большая скорость поиска? Всё очень просто, за счёт индексации. Так можно искать только по индексированным полям. Ну, хватит лишних слов, давай переходить к делу.

Для работы нам понадобятся всё те же таблицы из прошлого примера.

В качестве основы я взял только одну таблицу - mast.db. Как я уже говорил, для быстрого поиска нужно, чтобы поле было проиндексировано, поэтому добавь ещё один вторичный индекс для поля FIO (где у нас храниться фамилия). Теперь у компонента *TTable*, к которой привязана эта таблица (у меня это *Table1*), свойство *IndexFieldName* измени на *FIO*. Всё таблица готова к употреблению.

Теперь давай добавим на форму одну строку ввода, в которой будем вводить данные, которые надо искать и кнопку, по нажатию которой будет запускаться поиск. На рисунке 16.3.1 показана обновлённая форма нашей программы.

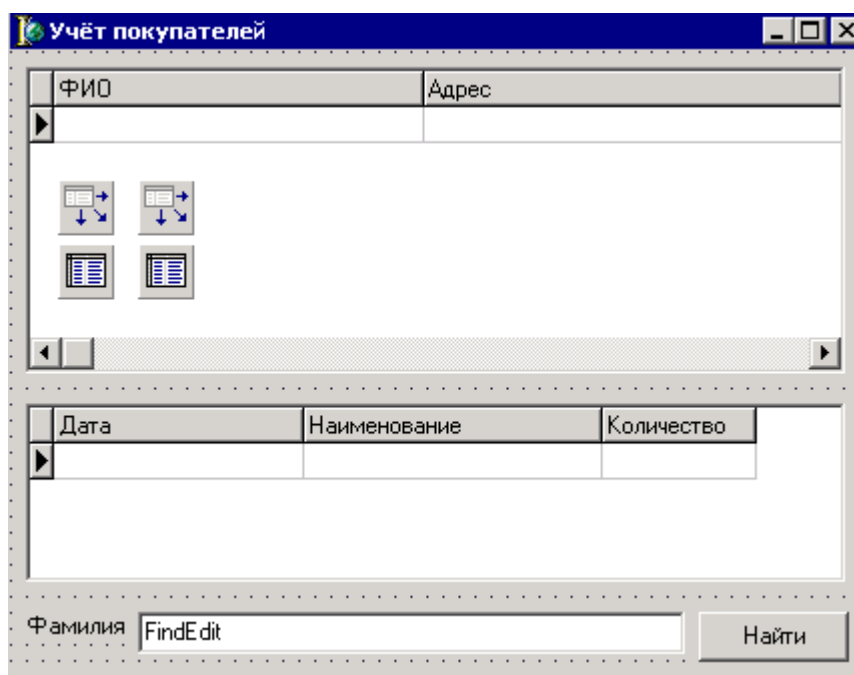


Рис 16.3.1. Обновлённая форма программы

Теперь создаём обработчик события *OnClick* для кнопки и пишем там следующий код:


```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Table1.SetKey;
  Table1FIO.AsString:=FindEdit.Text;
  Table1.GotoKey;
end;
```

Здесь всё очень просто. Первая строка говорит, что сейчас я буду устанавливать ключ. Вторая строка устанавливает его. Заметь, что это происходит как изменение содержимого поля, но поле не меняется, потому что мы вызвали перед этим *SetKey*. И, наконец, последняя строка говорит, что надо найти установленный ключ.

Всё время мы создавали одиночные вторичные ключи (состоящие из одного поля), но ты можешь с помощью DatabaseDesktop создавать ключи состоящие из любого количества полей. Для этого в DatabaseDesktop, в выпадающем списке *Table properties* нужно выбрать *Secondary Indexes* и нажать кнопку *Define*. Теперь выбери любое поле, и перемести его в список *Indexed fields*, затем другое поле и снова перемести его в список *Indexed fields*. И так хоть все поля.

Если у тебя установлен ключ из нескольких полей, то ты должен между вызовами *SetKey* и *GotoKey* установить все ключи. Иначе результат может быть ошибкой.

Есть ещё одна процедура, с которой я хочу тебя познакомить - *GotoNearest*. Если *GotoKey* не находит нужного ключа, то генерируется ошибка. *GotoNearest* тоже производит поиск ключевого поля, но если поле не найдено, то ошибка не генерируется, а ищется ближайший похожий ключ.

 На компакт диске, в директории \Примеры\Глава 16\FastFind ты можешь увидеть пример этой программы.

16.4 Создание псевдонимов.

В первом примере работы с таблицами paradox я показал тебе, как с помощью SQL Explorer можно создавать псевдонимы. Но что делать, если в твоей программе используются псевдонимы, а на компьютере пользователя их нет? Напрашивается ответ – создать их. А теперь представь себе, что пользователь живёт в другом городе и абсолютно ничего не понимает в компьютере. В этом случае, ты не сможешь объяснить ему процесс создания псевдонимов по телефону, а из этой ситуации нужно как-то находить выход. А выход простой – создать псевдонимы программно. Пуская твоя программа следит за наличием на компьютере пользователя псевдонимов.

Давай создадим маленькую программу, которая будет создавать какой-нибудь произвольный псевдоним таблиц. Запусти Delphi и создай новый проект.

Брось на форму компонент *TSession* с закладки BDE. Если ты работаешь с базой данных, то этот объект всегда создаётся автоматически без твоего ведома. В нём хранится низкоуровневая информация о BDE, драйверах и многом другом. Если ты хочешь изменить какие-то значения указанные в этом объекте по умолчанию, то тебе необходимо:

1. Поставить этот компонент на форму.
2. Указать любое имя в его свойстве *SessionName*.
3. Всем компонентам *TTable* и *TQuery*, для которых будут действовать установленные тобой значения, в поле *SessionName* нужно указать имя объекта *TSession*.

Таким образом ты сам создашь сессию.

Для нашего примера достаточно первых двух действий, потому что у нас не будет компонентов *TTable* или *TQuery*. Посмотри на рисунок 16.4.1 и сконструируй форму, подобную этой. На моей форме всего лишь две кнопки, один компонент *TListBox* и один компонент *TSession*. Ты можешь расположить их по другому, главное чтобы они присутствовали и тебе удобно было с ними работать.

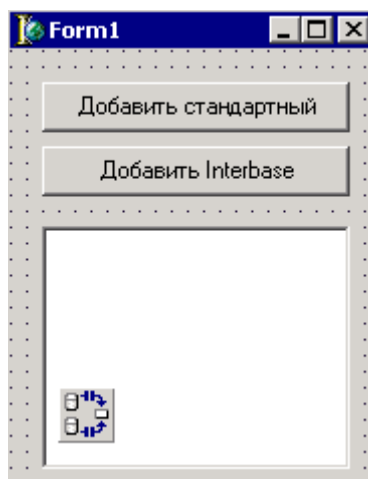


Рис 16.4.1. Обновлённая форма программы

Создай обработчик события *OnShow* для главной формы и напиши в нём следующее:

```
procedure TForm1.FormShow(Sender: TObject);
var
  str:TStrings;
begin
  Str:=TStringList.Create;
  Session1.GetAliasNames(Str);
  ListBox1.Items.Assign(Str);
  Str.Free;
end;
```

Здесь я получаю список всех доступных в системе псевдонимов и вставляю этот список в компонент *TListBox*. Рассмотрим построчно. Сначала я объявляю переменную *Str* типа *TStrings*.

Функция *GetAliasNames* компонента *TSession* возвращает все доступные системе псевдонимы. В качестве параметра передаётся указатель на объект типа *TStrings*, куда запишутся весь список.

ListBox1.Items.Assign(Str) - копирую список в элементы *ListBox1*.

И последняя строка освобождает переменную *Str*.

Теперь создадим новый псевдоним стандартного типа. К этому типу относятся таблицы *Paradox* и *DBF*. Я создаю псевдоним по нажатию первой кнопки. Вот соответствующая процедура:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  str:TStrings;
begin
  Session1.AddStandardAlias('VROnline','c:\','Paradox');
  Str:=TStringList.Create;
  Session1.GetAliasNames(Str);
  ListBox1.Items.Assign(Str);
  Str.Free;
end;
```

Создание происходит с помощью функции *AddStandardAlias* компонента *TSession*. В качестве первого параметра, ты должен указать имя нового псевдонима. Второй - путь к базам данных, которые будут связаны с псевдонимом. Третий - драйвер, который будет использоваться по умолчанию.

После добавления, я снова получаю уже обновлённый список доступных псевдонима и копирую его в компонент *TListBox*.

Теперь создадим псевдоним нестандартного типа. Для примера будет использоваться псевдоним для базы данных Interbase. Эти базы данных в книге рассматриваться не будут, но для примера я покажу, как создаются такие псевдонимы. Он будет создаваться по нажатию второй кнопки:

```
procedure TForm1.Button2Click(Sender: TObject);
var
  L: TStringList;
begin
  L := TStringList.Create;
  try
    with L do
      begin
        Add('SERVER NAME=IB_SERVER:/PATH/DATABASE.GDB');
        Add('USER NAME=MYNAME');
      end;
      Session1.AddAlias('NewIB', 'InterBase 4.x Driver by Visigen',L);
    finally
      L.Free;
    end;
  end;
end;
```

Процедура *AddAlias* компонента *TSession* создаёт псевдоним нестандартного типа. Первый параметр - имя псевдонима. Второй - имя драйвера. Третий параметр - указатель на *TStringList*, в котором хранятся дополнительные настройки. Дополнительные параметры хранятся в виде строк (например 'SERVER NAME= IB_SERVER:/PATH/DATABASE.GDB'). В строке указывается имя параметра и после знака равно его значение. Параметры могут отличаться, в зависимости от драйвера. Чтобы узнать, какие параметры доступны, можно создать пробный псевдоним в SQL Explorer и посмотреть, какие там присутствуют параметры.

Внимание!!! - имя драйвера у тебя может отличаться. Чтобы увидеть все доступные драйверы, нужно войти в *BDE Administrator*. Здесь, на закладке *Configuration* на дереве выбираешь *Drivers*, и здесь в разделах *Native* и *ODBC* есть все имена драйверов.

 На компакт диске, в директории \Примеры\Глава 16\Alias ты можешь увидеть пример этой программы.

16.5 Работа с XML таблицами.

Когда мы работаем с базами данных Access, то используем библиотеку DAO. Для доступа к таблицам Paradox или DBF, нужна библиотека BDE. Для XML таблиц нужен всего один файл, который достаточно зарегистрировать в системе и больше никакой головной боли. Из-за такой простоты установки, мы не сможем получить мощь других баз данных, зато получаем простоту, скорость и универсальность XML.

Итак, для того, чтобы ты мог работать с XML тебе понадобится файл *midas.dll*. Чаще всего он находится в системной директории *windows*. Для Windows 95/98/ME это *windows\system*, а для NT/2000/XP это *WinNT\system32*. Если у тебя этого файла нет, то можешь взять его с диска этой книги, в директории *dll/midas*. Там же находится файл *regsvr32.exe*, который может произвести регистрацию этого *dll* файла. Для регистрации нужно выполнить команду *regsvr32.exe* с параметром *midas.dll*. Например, помести эти файлы в систему *c:\windows\system* (*regsvr32.exe* уже может находиться там), и нажми кнопку «Пуск» и выбери «Выполнить». Здесь напиши следующее *c:\windows\system\regsvr32.exe c:\windows\system\midas.dll*. Затем нажми ОК и ты должен увидеть сообщение о успешной регистрации.

Создавать XML таблицу мы будем прямо в Delphi. Для этого создай новый проект и сразу же добавь к нему модуль данных. Теперь брось в модуль данных два компонента: *DataSource* и *ClientDataSet1*. Сразу же укажи у компонента *DataSource* в свойстве *DataSet* компонент *ClientDataSet*, чтобы связать их.

Теперь выделяй компонент *ClientDataSet*. Дважды щёлкни по свойству *FieldDefs* в объектном инспекторе чтобы открыть окно определения полей (рисунок 16.5.1). В этом окне можно создавать объявления новых полей. Для этого щёлкни правой кнопкой мышки и выбери пункт *Add*, чтобы создать новое поле. Выдели в окне строку нового поля и посмотри в объектный инспектор. Тут у нас есть три интересных свойства:

DataType – тип поля.

Name – имя.

Size – размер.

Первое поле у нас будет ключевым, поэтому выбирай тип поля *ftAutoInc* и имя *Key1*. Остальное не изменяем. Теперь создай ещё одно поле типа *ftString*, с именем *FIO* и размер поля оставим по умолчанию 20. Для примера этого хватит, хотя ты можешь создать ещё несколько полей.

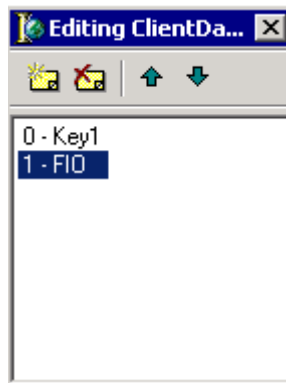


Рис 16.5.2. Обновлённая форма программы

Теперь щёлкни правой кнопкой по компоненту *ClientDataSet* и в появившемся меню выбери пункт *Create DataSet*. Теперь ещё раз щёлкни правой кнопкой по этому же компоненту и перед тобой откроется меню, в котором уже намного больше пунктов. Выбери *Save to MyBase Xml Table* и перед тобой откроется стандартное окно сохранения файла. Введи имя *exapl* и нажми «Сохранить».

Всё таблица готова. Теперь можно с ней работать абсолютно так же, как и с ADO или BDE таблицами. Дважды щёлкни по компоненту *ClientDataSet* чтобы открыть редактор полей. Здесь добавь все поля и дай им нормальные, русские заголовки. Кстати, ключевое поле можно вообще спрятать.


Теперь брось на главную форму программы сетку *DBGrid* и свяжи её с нашей таблицей. Программу можно запускать и попробовать с ней поработать. Здесь я не буду

писать полноценного примера и ограничусь только этим, потому что свойства и методы компонента *ClientDataSet* схожи с компонентами *TTable* и *TADOTable*.

Здесь я хочу сделать только одно замечание. Таблицы XML по умолчанию растут достаточно быстро. Это связано с тем, что в них сохраняется журнал изменений. С одной стороны постоянный рост файлов очень быстро есть пространство на диске, а с другой, благодаря этому журналу ты можешь в любой момент отменить последние действия. Для этого нужно вызвать метод *UndoLastChange* компонента *ClientDataSet*. У этого метода есть один параметр – булево значение. Если он равен *true*, то текущий курсор перебежит на строку, изменения которой были отменены, иначе курсор останется на месте.

Чтобы очистить журнал изменений вызови метод *MergeChangeLog*, а если хочешь, чтобы журнал вообще не вёлся, то присвой свойству *LogChanges* значение *false*.

Вот и всё, что я хотел сказать про XML таблицы. В остальном работа с ними ничем не отличается от работы с другими базами данных.

 На компакт диске, в директории \Примеры\Глава 16\XML ты можешь увидеть пример этой программы.