Delphi и базы данных

Работа с базами данных в Delphi реализована на самом высоком уровне, хотя этот язык и не создавался специально. Даже специализированные языки для работы с базами данных (такие, как MS Visual FoxPro) часто уступают по простоте и мощи программирования с использованием баз данных.

Для примеров будут использоваться базы Access и современный формат xml. Рекомендуется использовать эти базы в качестве локальных, потому что они поддерживаются большинством систем и отличаются высокой надёжностью.

Будут рассмотрены также самые простые и распространённые базы dbf и paradox. Использовать такие базы рекомендуется в тех случаях когда уже есть готовые базы, созданные в других приложениях Эти базы не рекомендуется использовать из-за их ненадёжности (в них регулярно нарушается индексная целостность, что приводит к неработоспособности программ). Но из-за их распространённости, знать принципы работы с ними просто необходимо. Так, например, локальная версия 1С Предприятия использует формат DBF.

Теория реляционных баз данных

Ещё десять лет назад, программирование баз данных было очень сложным занятием. За какие-либо достижения в этой области многие программисты получили в своё время докторские степени. Сейчас уже такое трудно себе представить, потому что благодаря Delphi, процесс написания программ упростился, а количество разновидностей баз данных уже исчисляется десятками.

Ключ	Фамилия	Имя	Отчество
1	Иванов	Иван	Петрович
2	Петров	Сергей	Владимирович
3	Сидоров	Алексей	Викторович
4	Смирнов	Андрей	Сергеевич

Пример простейшей базы данных

Базы данных делятся на **локальные** (установленные на компьютере клиента, там же где работает программа) и удалённые (установленные на сервере, удалённом компьютере). Серверные базы данных располагаются на удалённом компьютере и работают под управлением серверного программного обеспечения. К их главным преимуществам можно отнести возможность работы с одной базой данных одновременно несколькими пользователями, и при этом осуществляется минимальная нагрузка на сеть.

Есть ещё сетевые базы данных, но их мы рассматривать не будем, потому что они создают слишком большую нагрузку на сеть и неудобны в работе, как для программиста, так и для конечного пользователя. Поэтому работать с такими базами мы не будем. Почему?

Когда программа присоединяется к сетевой базе данных, то она выкачивает с сервера практически полную его копию. Если внёсти изменения, то копия полностью закачивается обратно. Это очень неудобно, потому что создаётся большая нагрузка на сеть из-за излишней перекачки данных.

При клиент-серверной технологии программа клиент посылает простой текстовый запрос на сервер на получение каких-либо данных. Сервер обрабатывает его и возвращает только необходимую порцию данных. Когда нужно изменить какие-то данные, опять посылается запрос к серверу на их изменение и сервер изменяет данные в своей базе. Таким образом, по сети происходит перекачка в основном только текстовых запросов, которые в основном занимают меньше килобайта. Все данные обрабатывает сервер, а значит, машина клиента загружается намного меньше и не так сильно требовательна к ресурсам. Сервер отсылает клиенту только самые необходимые данные, а значит, отсутствует излишняя перекачка копии все базы.

Благодаря всему этому, сетевые базы данных уже устарели и практически не используются. Их практически полностью вытесняет технология клиент-сервер. А вот локальные базы данных будут жить все-

гда. Может измениться формат их хранения или добавиться какие-то новые функции, но сами базы данных будут существовать.

Здесь рассмотрим только локальные базы данных, а серверные рассмотрим немного позже. Для дальнейшего рассмотрения надо определить новое понятие -таблица. Таблица базы данных - это двумерный массив, в котором в столбец выстроены данные (пример таблицы - Excel). База данных - грубо говоря это всего лишь файл, в котором может храниться от одной до нескольких таблиц. Большинство локальных баз данных могут хранить только одну таблицу (dBase, Paradox, XML). Но есть представители локальных баз, где в одном файле заключено несколько таблиц (например Access, который мы будем рассматривать в этой главе).

Локальные базы данных

Из локальных баз данных мы будем рассматривать реляционные, как самые распространённые. Что такое реляционная база данных? Это таблица, в которой в качестве столбцов выступают имена хранимых в ней данных, а каждая строка хранит сами данные. Таблица базы данных базы данных похожа на электронную таблицу Excel (если быть точнее, то Excel хранит свои данные в виде собственного формата, построенного на основе технологии баз данных). Локальные таблицы баз данных могут храниться на локальном жёстком диске или централизовано сохраняться на сетевой диск файлового сервера. Эти файлы можно копировать с помощью стандартных средств как любой другой файл, потому что сами таблицы базы данных не привязаны к определённому месту расположения. Главное, чтобы программа могла найти твою таблицу.

В каждой таблице должно быть одно уникальное поле, которое однозначно будет идентифицировать строку. Это поле называется ключевым. Эти поля очень часто используются для связывания нескольких таблиц между собой (с этим мы ещё познакомимся). Но даже если у тебя таблица не связана, ключевое поле всё равно обязательно. Представим, что пишется телефонная база данных. Вопрос "Сколько будет "Ивановых"? Как отличать их? Вот тут поможет ключ. В качестве ключа желательно использовать численный тип и если позволяет база данных, то будет лучше, если он будет типа "autoincrement" (автоматически увеличивающееся/уменьшающееся число или счётчик).

Имена столбцов в таблице базе данных, также должны быть уникальными, но в этом случае не обязательно числовыми. Их можно называть как угодно, лишь бы было уникально и понятно.

Каждый столбец (поле базы данных) обязательно должен иметь определённый тип. Количество типов и их разновидности зависят от типа базы данных, например формат dBASE (файлы с расширением DBF) поддерживает только 6 типов, а Paradox уже до 15.

База данных может храниться в одном файле (Access) или в нескольких (Paradox, dBase). Точнее сказать, данные таблицы всегда хранятся в одном файле, а вот дополнительная информация может располагаться в отдельных файлах. В качестве дополнительной информации

могут быть индексы, ограничения или список значений по умолчанию для конкретных полей. Если хотя бы один из файлов запортиться или будет удалён, то данные могут стать недоступными для редактирования.

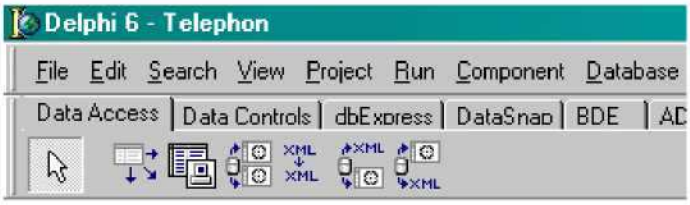
Что такое индексы? Очень часто данные из таблиц подвергаются каким-то изменениям, поэтому прежде чем произвести редактирование над какой-либо строкой, необходимо её найти. Даже статические таблицы, использующиеся в качестве справочников, тоже подвергаются операциям поиска перед выводом запрашиваемых данных. Поиск достаточно трудоёмкая операция, особенно если таблица содержит очень много строк. Индексы направлены на ускорение этой процедуры, а так же могут использоваться в качестве отправной точки при сортировке. На данном этапе достаточно знать, что не проиндексированное поле невозможно упорядочить.

Если необходимо, чтобы какая-то таблица была упорядочена по полю «Фамилия», то это поле надо сначала проиндексировать. Затем нужно только указать, что таблица должна работать сейчас с таким-то индексом, и она сортируется автоматически.

Для работы с базами в Delphi есть несколько наборов компонент. Каждый набор подходит для решения определённого круга задач. Почему такое разнообразие компонентов? Все они используют разные технологии доступа к данным и отличаются по возможностям. В отличие от Microsoft, которая встроила в свои продукты разработки только технологию доступа к данным ADO собственной разработки, фирма Borland дает разнообразие средств работающих через разные технологии и не ограничивается только своими разработками. Такой подход имеет свои преимущества.

Помимо этого есть группы, которые могут использоваться в любом случае. Дадим краткий обзор доступных средств.

На закладке Data Access расположены основные компоненты доступа к данным. Эти компоненты общие для всех и могут использоваться совместно с другими группами компонентов.

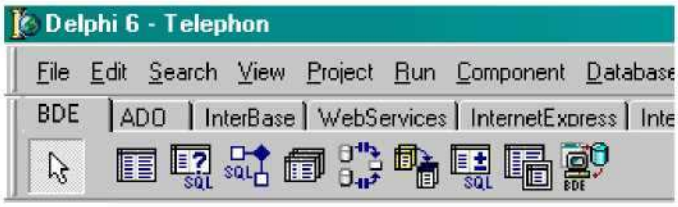


Закладка Data Access палитры компонентов

На закладке Data Controls расположены компоненты для отображения и редактирования данных в таблицах. Эти компоненты так же используются в не зависимости от используемой технологии доступа к данным.

Закладка **Data Controls** палитры компонентов

Закладка **BDE** содержит компоненты, позволяющие получить доступ к базам данных по технологии, разработанной фирмой **Borland** под названием **Borland Database Engine.** Эта технология сильно устарела и поставляется только для совместимости со старыми версиями. Не смотря на это, она хорошо работает со старыми типами баз данных, такими как **Paradox** и **dBase.**



Закладка **BDE** палитры компонентов

DBExpress - это новая технология доступа к данным фирмы **Borland.** Она отличается большей гибкостью и хорошо подходит для программирования клиент серверных приложений, использующих базы данных. Компоненты с одноимённой закладки я советую использовать с базами данных построенных по серверной технологии, например, **Oracle, DB2** или **MySQL.**

Закладка **dbExpress** палитры компонентов

ADO (Active Data Objects) - технология доступа к данным, разработанная корпорацией **Microsoft.** Очень хорошая библиотека, но рекомендуется использовать ее только с базами данных **Microsoft,** а именно **MS Access** или **MS SQL Server.** Её так же можно использовать для специфического сервер баз данных, который может работать только через **ODBC.**

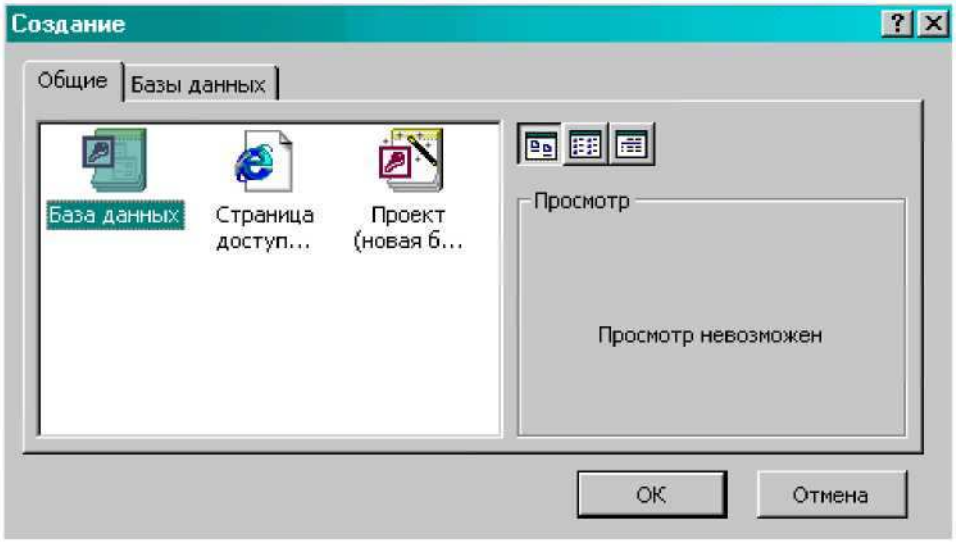
Закладка **ADO** палитры компонентов

Работа с базами данных Access идёт через специальную надстройку DAO, которая может устанавливаться на компьютер вместе с программой Office или идти как отдельная установка. Так что если созданная программа не будет работать на компьютере клиента, то надо позаботиться о установке DAO на этот компьютер.

Создание первой базы данных Access

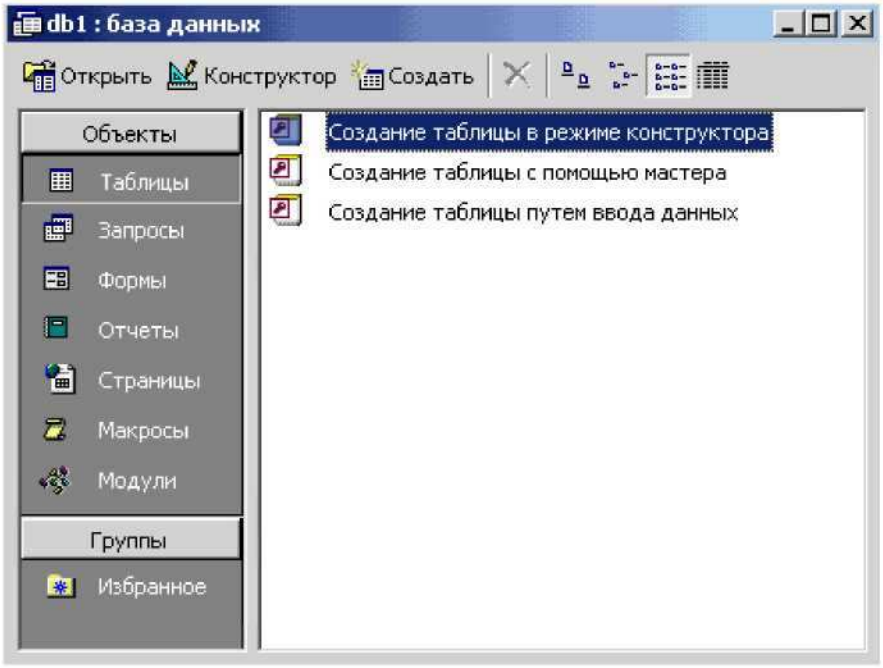
Для последующей работы необходимо, чтобы на компьютере был установлен MS Office и его компонент MS Access. Именно в нём и будут создаваться базы, а вот работать с ними будем из Delphi.

Запустите Access и выберите в меню Файл->Создать. В мастере создания базы выберите пункт "База данных" и нажмите "ОК" (см.рисунок) Будет предложено выбрать имя базы и место расположения, укажите что угодно, но назовите файл Database.mdb .



Окно создания новой базы данных

После этого Access создаст базу и сохранит её по указанному пути. Откроется окно (см.следующий рисунок), в котором и происходит работа с базой. С левой стороны окна находится колонка выбора объектов, с которыми можно работать. Первым находится пункт "Таблицы" (он выделен по умолчанию) который и будет нас интересовать. Если этот объект у не выделен, то надо выделить его.



Окно создания новой базы данных

В окне справа находится три пункта:

1. Создание таблицы в режиме конструктора
2. Создание таблицы с помощью мастера
3. Создание таблицы путём ввода данных

С помощью этих команд можно создать таблицы внутри созданной базы данных, Access, которая может хранить в одном файле несколько таблиц.

Все данные в базах данных хранятся в виде двухмерных таблиц. На сл.рисунке приведен пример простой базы данных, состоящей из семи колонок и множества строк.

Склад : таблица

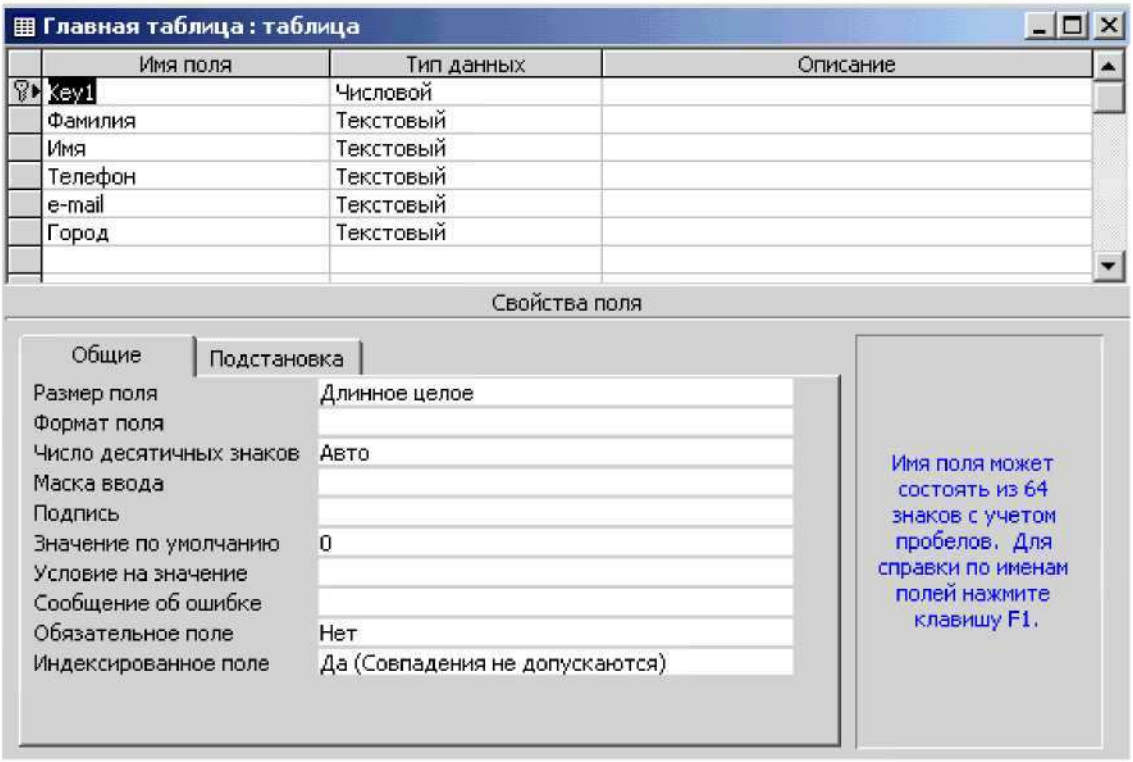
Код1	Код	Материал	Наименовани	Ед	Производите	Всего
1020	39000180	Автомат.выкл		ШТ		5
1025	39000150	Автомат.выкл.		ШТ		3
1026	39000160	Автомат.выкл.		ШТ		5
1027	39000170	Автомат.выкл.		ШТ		5
1028	39000190	Автомат.выкл.		ШТ		5
1029	29006654	Автоматич.вык		ШТ		10
1030	29006651	Автоматич.вык		ШТ		10
1031	29006649	Автоматич.вык		ШТ		27
1032	29006492	Автоматич.вык		ШТ		11
1033	71200610	Амортизатор 0		ШТ		6

Запись: 1019 из 1777

Пример простой таблицы.

Колонки в таблицах называются полями. Создадим базу данных телефонного справочника. Выберем "Создание таблицы в режиме конструктора" чтобы создать новую таблицу в базе данных. Откроется окно

(см.сл. рисунок).



Окно создания таблицы.

Сверху находится сетка, в которой вводятся поля таблицы, их тип и описание (последнее не обязательно). Если вписать в сетку имя нового поля и указать тип, внизу окна появляются свойства нового поля. В зависимости от типа поля изменяется и количество свойств. Перечислим основные:

Масксимальная длина поля. Для текстового поля размер не может быть больше 255. Если текст длинее, то надо использовать "Поле Memo".

Формат поля. Здесь ты можешь указать внешний вид данных. Например, поле может выглядить как "Yes/No" для логических полей, или например "mmyyyy" для поля даты.

Маска ввода. Здесь мы вводим маску, которая отвечает за отображение поляпри редактировании. Если ты щёлкнешь на кнопке с точками "..." в строке "Маска ввода", то увидишь мастер создания маски.

Значение по умолчанию. Не требует комментариев.

Обязательное поле. Если пользователь не введёт сюда значение, то появится сообщение об ошибке. Такое поле не может быть пустым.

Пустые строки. Похоже на предыдущий, потому что это поле тоже не может быть пустым.

Индексированное поле. Может быть неиндексированным, индексированным с допуском совпадений, и индексированным без допуска совпадений. Основной индекс всегда без допуска совпадений. Осталь-

ные желательно с допуском.

Сжатие Юникод - позволяет сжать данные в соответствии с Юникод.

Создадим шесть полей:

Имя поля - **Key1**. Тип - счётчик. Это у нас будет ключик. Размер поля -"Длинное целое". Индексированное поле - "Да (Совпадения не допускаются)".

Имя поля - **Фамилия**. Тип - текстовый. Размер поля - 50. Индексированное поле - "Да (Допускаются совпадения)".

Имя поля - **Имя**. Тип - текстовый. Размер поля - 50. Индексированное поле -"Да (Допускаются совпадения)".

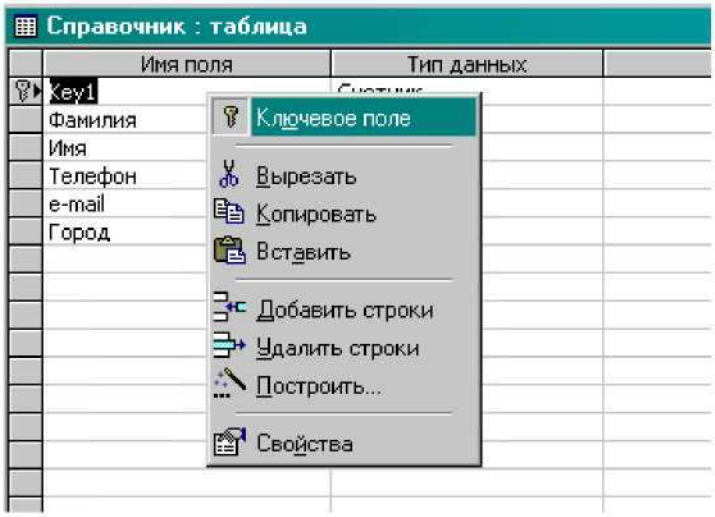
Имя поля - **Телефон**. Тип - текстовый. Размер поля - 10. Индексированное поле - "Да (Допускаются совпадения)".

Имя поля - **e-mail**. Тип - текстовый. Размер поля - 20. Индексированное поле -"Да (Допускаются совпадения)".

Имя поля - **Город**. Тип - числовой. Размер поля - Длинное целое. Индексированное поле - "Нет". Почему город не строковый, ведь названия городов - это текст? Пока не будем объяснять этот феномен.

Помимо этого, у всех полей значение "Обязательно поле" стоит в "Нет", и "Пустые строки" выставлено в "Да". Если сделать поле обязательным, то во всех строках обязательно должно быть заполнено соответствующее поле. Если запретить пустые строки (поставишь «Нет»), то в указанном поле должно быть обязательно что-то введено, иначе произойдёт ошибка. В реальных условиях, если какое-то поле обязательно должно иметь значение, то лучше сделать его обязательным. Не надо надеяться на добропорядочность пользователя. Пусть лучше база данных следит за правильностью данных.

Теперь выделите первое поле (Key1), щёлкните правой кнопкой мыши и выберите пункт "Ключевое поле" (см.рисунок). Задание ключевого поля является обязательным действием, если этого не сделать, то таблица не сможет редактироваться, а это значит, что в неё нельзя будет добавить строки.



Задание ключевого поля

Всё, теперь таблицу можно сохранять и закрывать. На вопрос: «Сохранить таблицу» отвечайте положительно и сохраняйте под именем «Справочник».

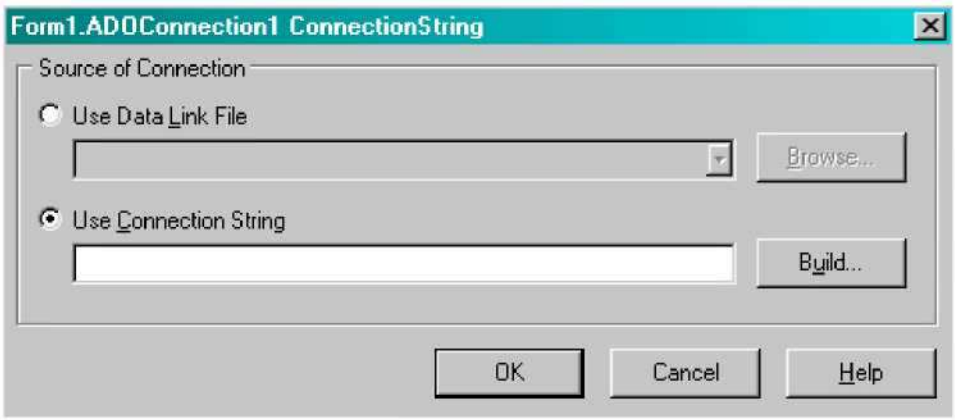
Первая база данных готова. Закрывайте её. Теперь перейдем к написанию примера для работы с базой и таблицей.

Пример работы с базами данных

Напишем программу, которая будет работать с базой данных MS Access. Для разработок с использованием MS лучше всего использовать **ADO**.

- ADOConnection.

Создадим новый проект. Теперь поместим на форму компонент ADOConnection с закладки ADO палитры компонентов. Теперь настроим соединение с сервером, которое должно быть прописано в свойстве ConnectionString. Для этого надо дважды щёлкнуть по строке ConnectionString и открывается окно, как на сл.рисунке

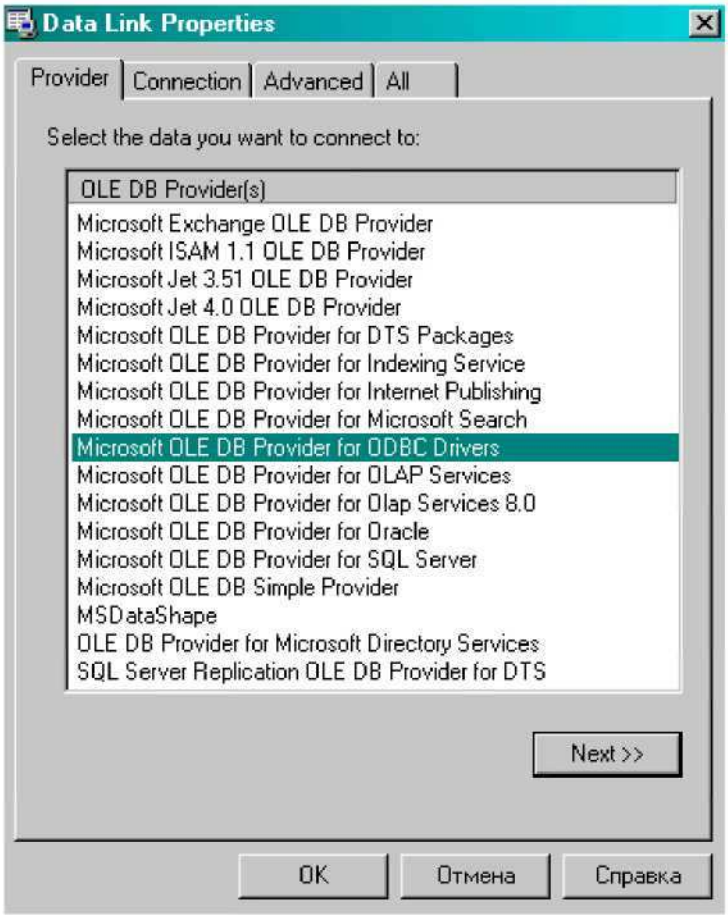


Окно создания подключения к базе

Здесь возможен выбор:

- Использовать специальный файл (Use Data Link File);
- Использовать строку подключения (Use Connection String).

Второе, как правило, более предпочтительно, поэтому рассмотрим, как создать строку подключения. Для этого щёлкнем кнопку Build и открывается ещё одно окно, показанное на сл.рисунке



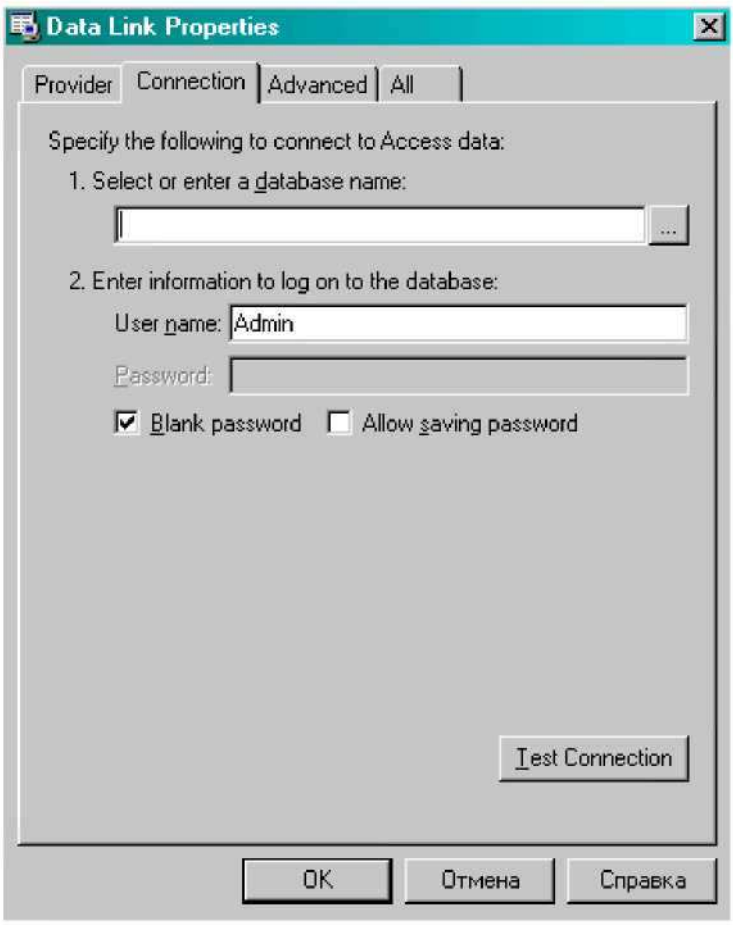
Окно создания строки подключения

На закладке Provider перечислены все доступные ADO драйверы доступа к базам данных. Если какого-то драйвера нет, то можно попробовать выделенный по умолчанию «Microsoft OLE DB Provider for ODBC Drivers». Этот драйвер позволяет получить доступ к базе данных через ODBC драйвер, которые есть к большинству существующих баз данных (единственное, он может быть не установленным на компьютере).

В нашем случае, для доступа к базам данных MS Access используется драйвер «Microsoft Jet OLE DB Provider». Такой драйвер обязательно устанавливается на машину вместе с MS Office, а в последних версиях Windows он устанавливается по умолчанию.

После этого нажимаем кнопку Next, или переходим на закладку «Connection».

Вид закладки Connection зависит от выбранного драйвера. В нашем случае она должна выглядеть, как показано на сл.рисунке:



Закладка Connection

Первым делом, в этом окне надо ввести имя (если надо то и путь) базы данных в строку «Select or enter a database name». Если база данных будет располагаться в той же директории, что и исполнимый файл, то путь указывать не надо. Рекомендуется хранить базы в одной директории с запускными файлами. Если держать файлы отдельно от исполнимого, то придётся указывать полный путь, а это может вызвать проблемы при переносе программы на другой компьютер. Ведь там программа будет искать базу по казанному пути, который может измениться. Если надо держать файлы в другой директории, то следует указывать относительный путь относительно текущей директории.

Чтобы легче было выбрать файл базы данных необходимо щёлкнуть по кнопке с точками справа от строки ввода.

Помимо этого нам надо заполнить следующие поля:

Имя пользователя **(User name)**, можно оставить по умолчанию, если незаданно иное при создании базы в MS Access;

Пароль **(Password)** - если база имеет пароль, то его необходимо указать;

Пустой пароль **(Blankpassword)** - если пароль не нужен, то здесь желательно поставить галочку;
Позволять сохранять пароль **(Allow saving password)**. Если здесь поставить галочку, то пароль может быть сохранён.

После выбора базы данных, надо нажать кнопку **Test Connection**, чтобы протестировать соединение. Если всё указано правильно, то должно появиться сообщение **«Test connection succeeded»**. Дальше, можно нажать ОК, чтобы закрыть окно создания строки подключения и ещё раз ОК, чтобы закрыть окно редактора строки подключения.

Теперь в свойствах компонента ADOConnection надо отключить свойство **LoginPrompt**, выставив его в False. Это нужно для того, чтобы при каждом обращении к базе нас не появлялось окно ввода пароля. А теперь выставим свойство **Connected** в **True**, чтобы произошло соединение с базой.

На этом соединение можно считать оконченным. Теперь надо получить доступ к созданной таблице «Справочник». Для этого поместим на форму компонент ADOTable с закладки ADO палитры компонентов. Сразу изменим его свойство Name на **BookName**.

 - **TADOTable**.

В этом компоненте тоже есть свойство **ConnectionString** и его так же можно настраивать. Почему «можно»? Да потому что, чтобы этого не делать мы поставили на форму компонент ADOConnection. Теперь мы можем указать у нашего компонента BookName в свойстве Connection, созданный нами компонент соединения с базой данных. Щёлкнем по выпадающему списку в свойстве Connection и выберем там единственный пункт ADOConnection1. Теперь не надо заполнять свойство ConnectionString.

Теперь в свойстве **TableName** нужно выбрать имя нашей таблицы (Справочник). Всё, таблица и соединение указаны, можно подключатся. Для этого выставь свойство **Active** в **true**.

 - **TDataSource**

Для отображения данных из таблицы надо ещё установить на форму компонент **DataSource** с закладки Data Access палитры компонентов. Теперь этому компоненту надо указать, какую именно таблицу он должен отображать. Для этого в свойстве **DataSet** нужно из выпадающего списка выбрать нашу таблицу **BookTable**.

 **-DBGrid**

Всё!!! Все приготовления готовы, можно приступать к реальному отображению данных. Самый простой способ отобразить таблицу - установить компонент **DBGrid**. Это компонент-сетка, которая может отображать данные в виде таблицы. В этом же компоненте можно до-

бавлять, удалять и редактировать строки нашей таблицы.

Форма нашего приложения

И последний этап создания приложения - связывание компонента сетки с компонентом отображения таблицы. Для этого в свойстве **DataSource** компонента DBGrid нужно указать созданный нами компонент **DataSource1**.

Теперь приложение готово! Мы не написали ни одной строчки кода.

Запустим пример и создадим несколько строк, попробуем отредактировать уже существующие и удалить что-нибудь. Для вставки строки используется клавиша Ins, а для удаления Ctrl+Del.

Свойства компонента TADOTable

Компонент TADOTable имеет множество полезных свойств. Большинство из них просты в использовании, поэтому чтобы не писать множество примеров на их использование, кратко опишем основные из них.

MasterSource - в этом свойстве указывается главная, по отношению к текущей таблица.

ReadOnly - если это свойство равно true, то таблицу нельзя редактировать. В этом случае данные только отображаются. Обязательно устанавливайте это свойство для тех таблиц, где данные не должны изменяться и пользователь не должен вносить в них изменения.

TableDirect - это свойство отображает каким будет доступ к таблице. Если этот параметр равен true то будет прямой доступ к таблице по имени. Если false то будет специальный SQL запрос к базе данных. Не все базы данных позволяют работать через прямой доступ, поэтому это свойство по умолчанию равно false.

TableName - имя таблицы, данные которой мы хотим обрабатывать.

CacheSize - размер кэш памяти. Если здесь установить число 50, то при первом подключении к таблице компонент выберет первые 50 строк и поместит их в локальной памяти, что ускорит доступ к ним.

CanModify - свойство похоже на ReadOnly и указывает на возможность редактирование данных таблицы.

CommandTimeout - время ожидания выполнения команды. Когда компонент направляет команду базе данных, то он запускает таймер ожидания, по истечению которого (если команда не выполнилась) происходит сообщение об ошибке.

Connection - здесь указывается компонент TADOConnection, через который происходит подключение.

ConnectionString - строка подключения к базе данных.

CursorLocation - расположение курсора, который считывает данные и указывает текущую позицию в таблице. Курсор может находиться на сервере или на машине клиента.

CursorType - тип курсора. Тут возможен один из следующих вариантов:

ctUnspecified расположение курсора не указано

ctOpenForwardOnly - курсор может двигаться только вперёд.

ctKeyset при этом курсоре изменения внесённые одним пользователем не видны остальным пользователям подключённым к этой таблице. Если с одной таблицей работают одновременно несколько пользователей, то при таком курсоре для отображения изменений других пользователей нужно отключиться от базы и подключиться к ней снова.

ctDynamic динамический курсор, при котором изменения одного пользователя видят все остальные.

ctStatic статический курсор. Изменения одного пользователя не видны остальным

Внимание!!! Если курсор расположен на клиенте, то можно использовать только статический курсор. Не все типы курсоров могут работать с определённой базой данных. Одна база данных может поддерживать один тип, а другая может поддерживать всё.

Filter - строка фильтра.

Filtered - является ли таблица фильтруемой. Если здесь установить false то строка фильтра (filter) игнорируется.

IndexFieldNames - имя индексированной колонки. Индексы используются для сортировки данных или для связи между таблицами.

RecNo - номер текущей выделенной строки.

RecordCount - количество строк в таблице.

Sort - строка, в которой указывается тип сортировки. Например, для сортировки по полю «Телефон» сюда нужно записать строку: ADOQuery1.Sort := 'Телефон ASC'. Оператор ASC говорит о том, что

надо сортировать в порядке возрастания. Оператор DESC говорит о сортировании в порядке убывания.

Active - если это свойство равно true, то таблица открыта.

AggFields - здесь хранятся все агрегатные поля.

AutoCalcFields - если здесь true, то надо автоматически пересчитывать поля.

Bof - на это свойство влиять нельзя, но если оно равно true, то мы находимся в начале файла.

Bookmark - здесь находиться текущая закладка.

Eof- на это свойство влиять нельзя, но если оно равно true, то мы находимся в конце файла.

FieldCount - здесь хранится количество полей в таблице.

Fields - через это поле можно получить доступ к значениям полей. Допустим, что надо узнать, какое значение храниться в 4-м поле. Для этого нужно написать Table.Fields.Fields[4].AsString. Метод AsString говорит о том, что нам надо получить значение в виде строки.

FieldValues - с помощью этого свойства можно легко получить доступ к любому значению указанного поля. Имя поля нужно указывать в квадратных скобках.

Пример, Table1.FieldValues['Телефон']: = '3346598';

FilterOption - настройки фильтра. Здесь можно указывать следующие параметры:

foCaseInsensitive фильтр будет не чувствителен к регистру.

foNoPartialCompare если стоит этот параметр, то сравнения будут происходить с точной копией указанного значения в фильтре. Если параметр не казан, то в фильтр будут попадать строки содержащие значение в фильтре, но не являющиеся его точной копией. Например, если в фильтре указано показывать слова «са», то в фильтр попадут все слова начинающиеся на «са» (самолёт, самокат).

Modified - если это свойство равно true, то в таблице были внесены изменения.

Методы компонента TADOTable

BookmarkValid - этот метод проверяет правильность закладки. В качестве единственного параметра нужно указать закладку типа TBookmark и если она является действительной, то результатом бу-

дет true.

CancelUpdates - отменить обновления сохранённые в кэш памяти

CompareBookmarks.- сравнение двух закладок. У метода два параметра типа TBookmark. Эти две закладки сравниваются. Если закладки равны, то результат равен нулю. Если первая меньше второй, то результат будет -1. Если первая больше второй, то результат равен единице.

DeleteRecords - удалить записи. У метода один параметр - какие записи удалять. Параметры:

arCurrent удалить только текущую запись.

arFiltered удалить записи удовлетворяющие установленному фильтру.

arAll - все записи.

arAllChapters удалить записи во всех разделах ADO.

Append - добавить новую запись в конец таблицы.

Cancel - отменить изменения текущей строки, если изменения ещё не были сохранены с помощью метода Post.

Close - закрыть таблицу.

Delete - удалить текущую строку.

Edit - перейти в режим редактирования. После этого можно изменять значения полей.

FieldByName - найти поле по имени. В качестве единственного параметра нужно указать имя поля виде строки и в результате получаем ссылку на поле в виде объекта TField.

First - перейти на первую строку в таблице.

Insert - вставить новую строку в таблицу.

IsEmpty - если метод вернёт true то в таблице нет записей.

Last - перейти на последнюю запись в таблице.

Next - перейти на следующую запись.

Post - принять все изменения.

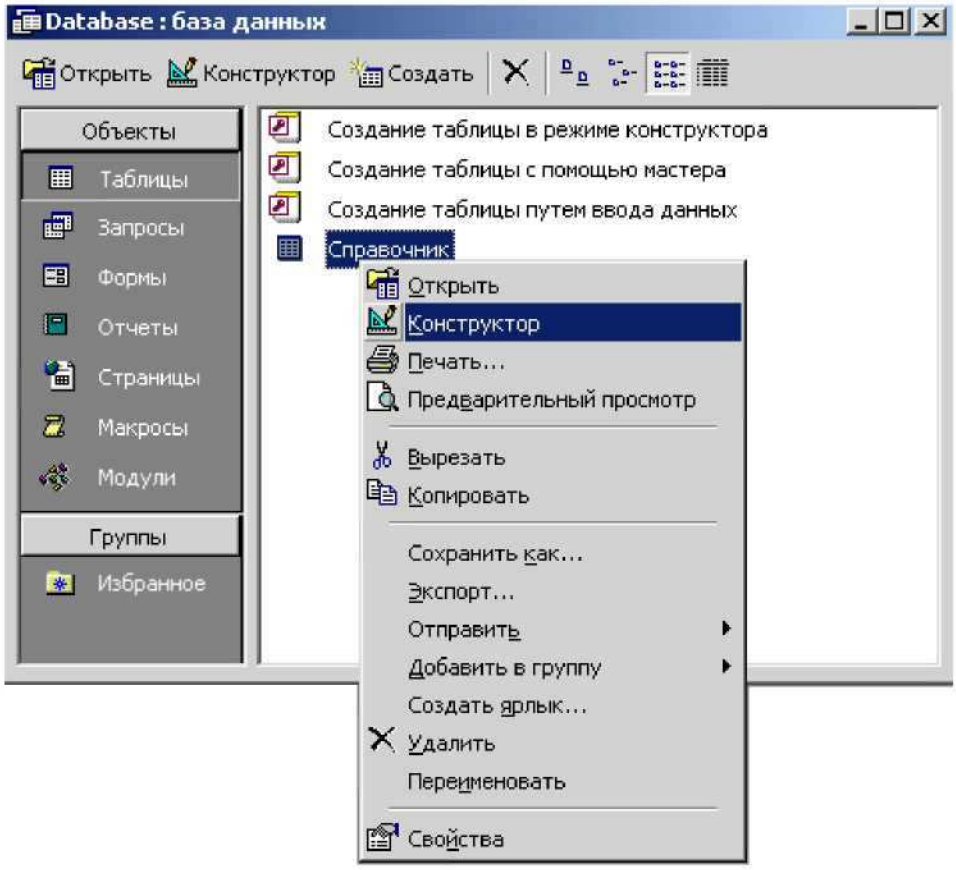
Prior - двигаться на предыдущую запись в таблице.

Refresh - обновить информацию о данных.

UpdateRecord - обновить текущую запись.

Управление отображением данных

Чтобы прятать от пользователя не нужные поля и показывать только то, что мы хотим и в том виде, в котором хотим, необходимо научиться управлять отображением данных. Но прежде чем приступить к этому, создадим в базе ещё два поля «Дата» и «Мобильник». Загрузим базу данных в Access, щёлкнем по ней правой кнопкой и в появившемся меню выберем «Конструктор».



Редактирование таблицы

Добавим поле с именем «Дата», тип «Дата/время».

Добавим поле с именем «Мобильник», и тип «Логический». Если в строке находится мобильный телефон, то в этом поле будем ставить true, иначе «false».

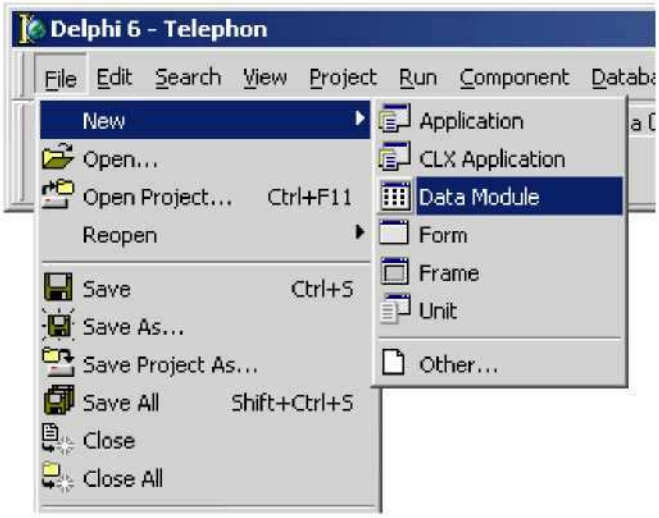
Закройте таблицу. Теперь переходим в Delphi и попробуем отобразить изменения в уже созданном примере.

Для начала перенесём компоненты доступа к базе данных в отдельное специальное окно.

Выделим компоненты ADOConnection1, DataSource1 и BookTable. Теперь выберем из меню Edit пункт Cut, что бы эти компоненты скопировались в буфер обмена и сразу удалились с формы.

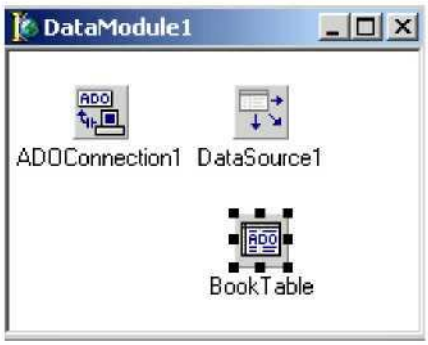
Теперь выберем из меню **File->New->Data Module** (см.сл.рисунок).

Delphi создаст специальное окно Data Module, которое удобно подходит для хранения компонентов доступа к базам данных.



Создание модуля Data Module

Теперь выберем из меню Edit пункт Paste, чтобы вставить в это окно вырезанные ранее компоненты. Расположим теперь эти компоненты в окне так, чтобы было удобно (см.сл.рисунок).

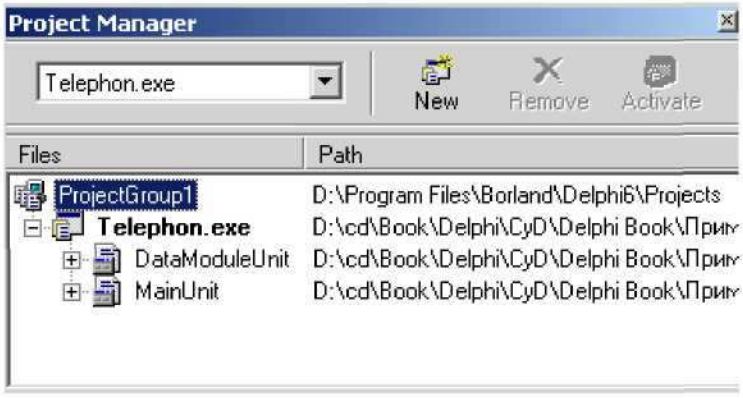


Окно Data Module

Теперь все компоненты, которые предназначены для доступа к базе данных будем располагать здесь, чтобы с ними удобно было работать. Сохраним новый модуль под именем DataModuleUnit.

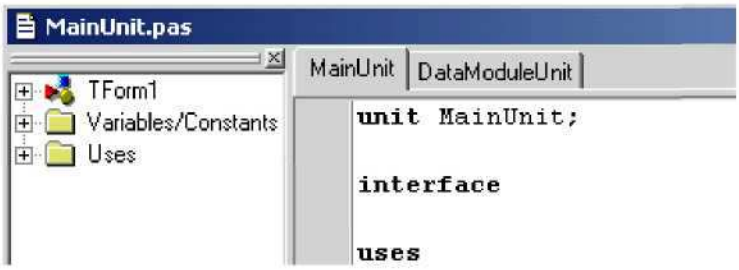
Теперь откройте менеджер проектов (в меню Project надо выбрать Project Manager) и расположим это окно так, чтобы было удобно в любой момент получить к нему доступ (например, в правом нижнем углу экрана).

Теперь, когда надо перейти из главной формы в модуль данных DataModule или обратно, можно сделать это с помощью менеджера проектов, дважды щёлкнув по нужной форме.



Менеджер проектов

Если хоть раз уже открывалась какая-то форма из менеджера проектов и не была закрыта, то её можно найти на закладках в окне редактора кода:

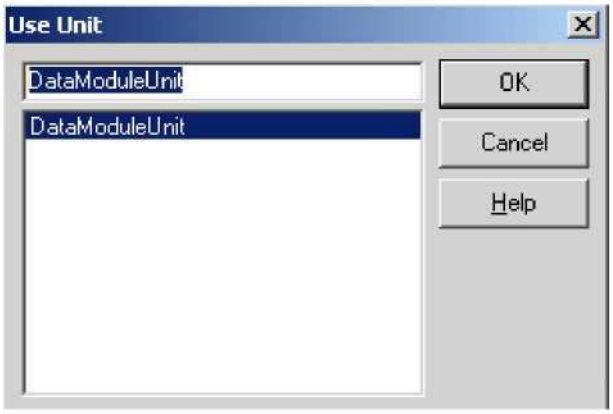


Закладки форм в редакторе кода

Напомним, что переключаться между визуальной формой и её кодом очень удобно простым нажатием клавиши F12.

Перейдем теперь в главную форму. В сетке DBGrid1 нет данных. Почему? Причина в том, что она потеряла связь с компонентами доступа к данным. Выделим сетку и щёлкнем по свойству DataSource,- в выпадающем списке ничего нет. Это потому что все нужные компоненты мы убрали в отдельную форму и главная форма пока об этом не знает.

Чтобы форма узнала о существовании компонентов, ей нужно указать в разделе uses наш модуль DataModuleUnit. Это можно сделать вручную или выбрать из меню File пункт Use Unit (в этот момент должно быть выделено окно кода главной формы, потому что мы подключаем новый модуль именно к ней. В появившемся окне (см.сл.рисунок) нужно выбрать имя нашего нового модуля DataModuleUnit (пока оно одно в списке) и нажать OK



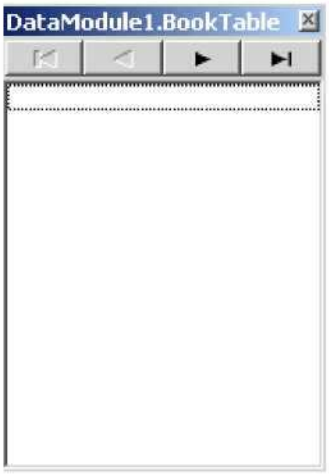
6 Окно подключения нового модуля

Проверим теперь в редакторе кода, чтобы после ключевого слова implementation появилось «uses DataModuleUnit;»:

```
implementation
uses DataModuleUnit;
{$R *.dfm}
```

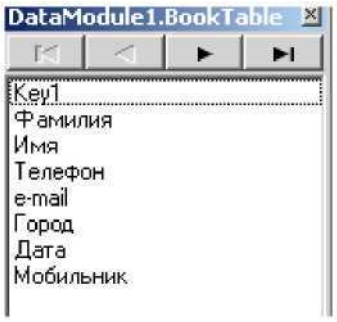
Теперь можно выделять нашу сетку DBGrid1 и в свойстве DataSource указывать компонент DataSource, данные которого должны быть отображены в сетке (DataModule1. DataSource1).

Теперь переходим в модуль DataModule чтобы настроить отображение данных. Дважды щёлкнем по компоненту BookTable-появиться окно редактирования полей базы данных (см.сл.рис.).



Окно редактирования полей базы данных.

Пока оно пустое и в него надо добавить все поля базы данных. Для этого щёлкнем в нём правой кнопкой мыши и в появившемся меню выберем пункт Add All Field (Добавить все поля). Окно автоматически заполниться именами полей (см.сл.рис.).



Заполненное окно редактирования
полей базы данных

Поля можно переставлять местами двигая мышкой. При этом физическое расположение в базе данных не меняется, зато при отображении данных в сетке, они будут отображаться в том порядке, в котором они выстроены здесь. Можно теперь выделять отдельные поля и в объектном инспекторе редактировать его свойства. Свойства у полей могут быть разные, в зависимости от типа поля.

Первое, что надо сделать - убрать из видимости счётчик (поле Key). Выделим это поле и в объектном инспекторе установим в свойстве Visible значение false (это свойство есть у всех полей). Сразу можно перейти в главную форму или запустить программу, чтобы убедиться в том, что поле Key1 больше не отображается.

Теперь отредактируем длину отображения колонок. Для этого выделим «Фамилия». В базе данных раньше мы выделили под это поле 50 символов (на всякий случай). В сетке ширина колонки будет отображаться по умолчанию на всю длину. Но чаще всего фамилии не превышают 15 символов, поэтому нет смысла отображать всю длину. На много удобней отображать только 15 символов, а если что-то не поместиться, то пользователь программы в любой момент сможет раздвинуть колонку и увидеть недостающие символы.

За ширину колонки отвечает свойство DisplayWidth (это свойство есть у всех полей). По умолчанию в нём стоит значение физической ширины поля, но мы укажем там 15. Опять же, на саму базу данных это не влияет и поле всё ещё имеет размер 50, но ширина отображаемой колонки в сетке будет 15. Сократим и ширину поля «Имя».

Ещё несколько интересных свойств:

DefaultExpression - здесь можно указать значение по умолчанию. В дальнейшем, когда будут создаваться новые строки, то в поля будут сразу заноситься указанные здесь значения.

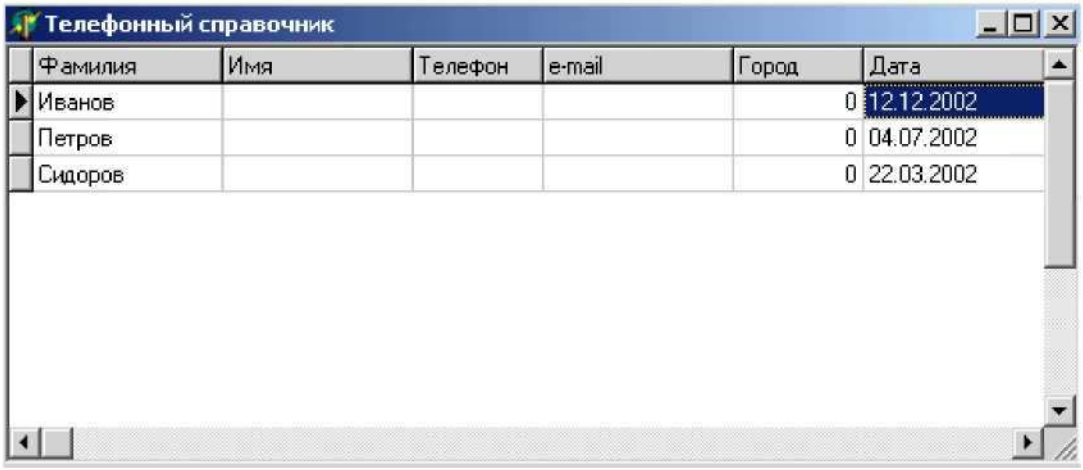
MaxValue - максимально допустимое значение. Если это числовое поле и оно должно изменяться в определённых рамках (например, от 0 до 100), то желательно указать эти ограничения здесь, чтобы сократить вероятность опечатки пользователем.

MinValue - минимально допустимое значение.

ReadOnly -поле только для чтения. Если какое-то поле не должно изменяться, то установим в свойстве ReadOnly значение true.

Required - если здесь true, то поле является обязательным и обязательно должно иметь какое-то значение. Если пользователь ничего не укажет, то программа сообщит об этом. Допустим, что какое-то поле участвует в расчётах. Если в этом поле не окажется данных, то программа может зависнуть. Есть два пути - при расчёте проверять наличие в поле данных или требовать, чтобы пользователь обязательно что-то ввёл. Второй путь предпочтительней, если это поле действительно важное. Представьте запись в телефонном справочнике без телефона. Спрашивается, зачем тогда нужна эта запись если не указан телефон. Так что поле для номера телефона можно делать обязательным.

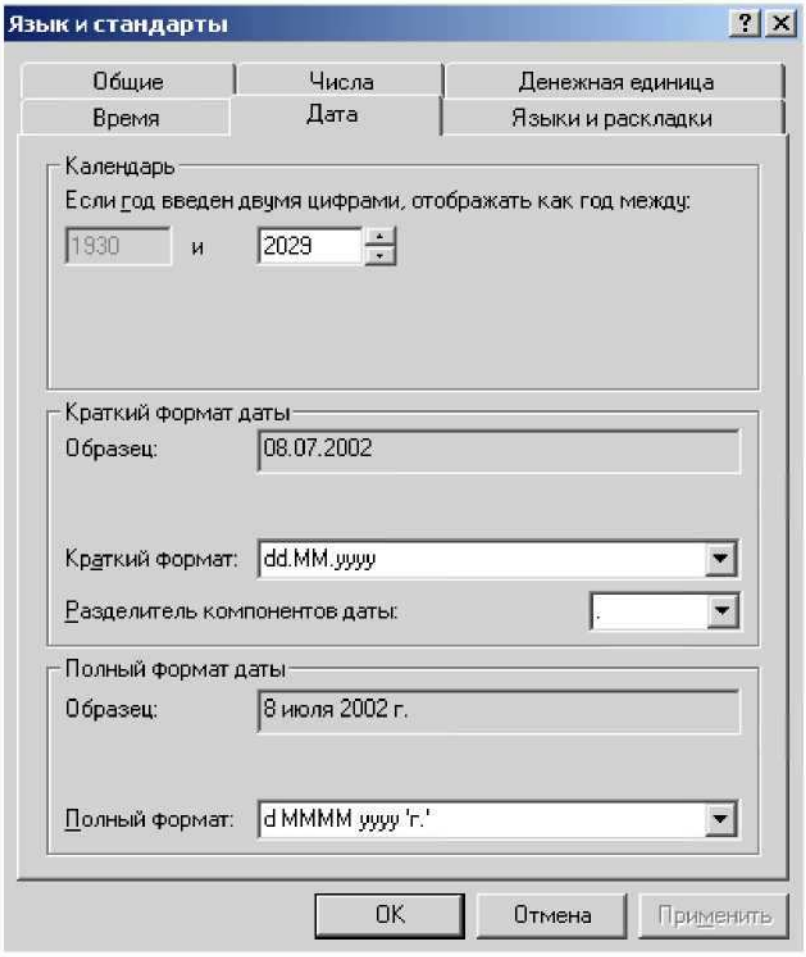
Tag- числовое значение, можно использовать по своему усмотрению.



Улучшенное окно программы.

Запустим программу и заполним поле «Дата» у всех записей любыми значениями. При заполнении надо указывать реальные даты. Если ввести недопустимое значение, то появится ошибка.

При вводе данных следует учитывать разделитель чисел. В большинстве русскоязычных ОС Windows в качестве разделителя используется точка или знак косой черты «/». К тому же первым идёт число, потом месяц и потом год (в англоязычной версии первым может идти месяц). Пробелы не допустимы. Этот порядок и разделитель настраиваются в настройках ОС. Войдите в «Панель управления» и запустите окно «Язык и стандарты» (см.сл.рисунок). Здесь можно изменить все настройки ввода даты, времени и чисел.



Настройка формата ввода и отображения даты

Вернёмся к Delphi. Выделим поле «Дата». Первое, что мы должны сделать -уточнить, какую именно дату здесь надо вводить. Так как это телефонный справочник, то можно, например, указывать дату рождения владельца телефона. Поэтому, вполне разумно будет в заголовке сетке отображать не просто «Дата», а «Дата рождения». Тут можно поступить двумя способами - отредактировать имя поля в базе данных (не совсем разумно) или просто заставить Delphi отображать в заголовке поля нужный текст.

За текст отображаемый в заголовке отвечает свойство DisplayLabel (это свойство есть у всех полей). Введем в нём текст «Дата рождения». Можно проверить, что теперь в заголовке отображается нужный текст.

Теперь отредактируем формат отображения даты. За это отвечает свойство DisplayFormat. Тут можно указывать текстовый формат, в котором нужно отображать дату. Как отображать? Вспомним функцию FormatDateTime и её первый параметр. Обычно используют для отображения полный формат - «dddddd».

Ну и наконец нужно указать маску ввода для даты. Её нужно указывать в свойстве EditMask так же как у компонента TMaskEdit. Для даты обычно указывают маску ввода «99/99/9999».

Если запустить пример, то в поле «Дата рождения» все даты будут

отображаться в полном формате (см.сл.рис.). Если щёлкнуть дважды по этому полю в любой строке (войти в режим редактирования строки), то дата сразу перейдёт в режим редактирования (см.сл.рис., вторая строка).

Улучшенный вид поля «Датарождения»

Последнее, что надо отредактировать - поле «Мобильник». Пока что здесь отображаются значения `true` или `false`, но будет удобнее видеть Да или Нет. Выделим это поле и в объектном инспекторе найдем свойство `Display Values`. Для булевых полей здесь нужно указать два значения в формате «`True;False`», т.е. сначала указываем положительное значение и после точки с запятой отрицательное (кавычки указывать не надо). Укажет Да;Нет.

Теперь в поле «Мобильник» будут отображаться понятные слова, да и при редактировании теперь нужно вводить не `true` или `false`, а Да или Нет.