Целью моей курсовой работы является программирование с помощью Delphi 5. Прежде чем начать выполнять свое задание рассмотрим эту систему программирования, в общем**.**

**Delphi** - это греческий город, где жил дельфийский оракул. И этим именем был назван новый программный продукт с феноменальными характеристиками.

Hадо отметить, что к моменту выхода продукта обстановка вокруг компании Borland складывалась не лучшим для нее образом. Поговаривали о возможной перепродаже компании, курс акций компании неудержимо катился вниз. Сейчас уже можно без всяких сомнений утверждать, что период трудностей позади. Hеверно, конечно, было бы говорить, что только **Delphi** явился причиной восстановления компании; кроме **Delphi**, у Borland появились и другие замечательные продукты, так же, как и **Delphi**, основывающиеся на новых, появившихся недавно у компании Borland, технологиях. Я имею в виду новые BDE 2.0, BC++ 4.5, Paradox for Windows 5.0, dBase for Windows 5.0, BC++ 2.0 for OS/2.

Тем не менее, именно **Delphi** стал тем продуктом, на примере которого стало ясно, что у Borland есть еще порох в пороховницах, и что один единственный продукт может настолько удачно сочетать несколько передовых технологий.

**Delphi** - это комбинация нескольких важнейших технологий:

1. Высокопроизводительный компилятор в машинный код
2. Объектно-ориентированная модель компонент
3. Визуальное (а, следовательно, и скоростное) построение приложений из программных прототипов
4. Масштабируемые средства для построения баз данных

Компилятор, встроенный в **Delphi**, обеспечивает высокую производительность, необходимую для построения приложений в архитектуре “клиент-сервер”. Этот компилятор в настоящее время является самым быстрым в мире, его скорость компиляции составляет свыше 120 тысяч строк в минуту на компьютере 486DX33. Он предлагает легкость разработки и быстрое время проверки готового программного блока, характерного для языков четвертого поколения (4GL) и в то же время обеспечивает качество кода, характерного для компилятора 3GL. Кроме того, **Delphi** обеспечивает быструю разработку без необходимости писать вставки на Си или ручного написания кода (хотя это возможно).

В процессе построения приложения разработчик выбирает из палитры компонент готовые компоненты как художник, делающий крупные мазки кистью. Еще до компиляции он видит результаты своей работы - после подключения к источнику данных их можно видеть отображенными на форме, можно перемещаться по данным, представлять их в том или ином виде. В этом смысле проектирование в **Delphi** мало чем отличается от проектирования в интерпретирующей среде, однако после выполнения компиляции мы получаем код, который исполняется в 10-20 раз быстрее, чем то же самое, сделанное при помощи интерпретатора. Кроме того, компилятор компилятору рознь, в **Delphi** компиляция производится непосредственно в родной машинный код, в то время как существуют компиляторы, превращающие программу в так называемый p-код, который затем интерпретируется виртуальной p-машиной. Это не может не сказаться на фактическом быстродействии готового приложения.

Основной упор в модели **Delphi** делается на максимальном реиспользовании кода. Это позволяет разработчикам строить приложения весьма быстро из заранее подготовленных объектов, а также дает им возможность создавать свои собственные объекты для среды **Delphi**. Никаких ограничений по типам объектов, которые могут создавать разработчики, не существует. Действительно, все в **Delphi** написано на нем же, поэтому разработчики имеют доступ к тем же объектам и инструментам, которые использовались для создания среды разработки. В результате нет никакой разницы между объектами, поставляемыми Borland или третьими фирмами, и объектами, которые вы можете создать.

В стандартную поставку **Delphi** входят основные объекты, которые образуют удачно подобранную иерархию из 270 базовых классов. Для начала - неплохо. Но если возникнет необходимость в решении какой-то специфической проблемы на **Delphi**, советуем, прежде чем попытаться начинать решать проблему “с нуля”, просмотреть список свободно распространяемых или коммерческих компонент, разработанных третьими фирмами, количество этих фирм в настоящее время превышает число 250, хотя, возможно, я не обо всех знаю. Скептики, возможно, не поверят мне, когда я скажу, что на **Delphi** можно одинаково хорошо писать как приложения к корпоративным базам данных, так и, к примеру, игровые программы. Тем не менее, это так. Во многом это объясняется тем, что традиционно в среде Windows было достаточно сложно реализовывать пользовательский интерфейс. Событийная модель в Windows всегда была сложна для понимания и отладки. Но именно разработка интерфейса в **Delphi** является самой простой задачей для программиста.

Cреда **Delphi** включает в себя полный набор визуальных инструментов для скоростной разработки приложений (RAD - rapid application development), поддерживающей разработку пользовательского интерфейса и подключение к корпоративным базам данных. VCL - библиотека визуальных компонент, включает в себя стандартные объекты построения пользовательского интерфейса, объекты управления данными, графические объекты, объекты мультимедиа, диалоги и объекты управления файлами, управление DDE и OLE. Единственное, что можно поставить в вину **Delphi**, это то, что готовых компонент, поставляемых Borland, могло бы быть и больше. Однако, разработки других фирм, а также свободно распространяемые программистами freeware-компоненты уже восполнили этот недостаток. Постойте, - скажете вы, ведь это уже было. Да, это было в Visual Basic.

Соответствующий стандарт компонент назывался VBX. И этот стандарт так же поддерживается в **Delphi**. Однако, визуальные компоненты в **Delphi** обладают большей гибкостью. Вспомним, в чем была проблема в VB. Прикладной программист программировал, вообще говоря, в среде языка бэйсик. А компоненты в стандарте VBX готовили ему его коллеги-профессионалы на С++.

VBX’ы приходили, “как есть”, и ни исправить, ни добавить ничего было нельзя.

А для изготовления VBX надо было осваивать “кухню” языка C++. В **Delphi** визуальные компоненты пишутся на объектном паскале, на том же паскале, на котором пишется алгоритмическая часть приложения. И визуальные компоненты **Delphi** получаются открытыми для надстройки и переписывания. Чувствуете разницу?

Объекты БД в **Delphi** основаны на SQL и включают в себя полную мощь Borland Database Engine. В состав **Delphi** также включен Borland SQL Link, поэтому доступ к СУБД Oracle, Sybase, Informix и InterBase происходит с высокой эффективностью. Кроме того, **Delphi** включает в себя локальный сервер Interbase для того, чтобы можно было разработать расширяемые на любые внешние SQL-сервера приложения в офлайновом режиме. Разработчик в среде **Delphi**, проектирующий информационную систему для локальной машины (к примеру, небольшую систему учета медицинских карточек для одного компьютера), может использовать для хранения информации файлы формата **.dbf** (как в dBase или Clipper) или **.db** (Paradox). Если же он будет использовать локальный InterBase for Windows 4.0 (это локальный SQL-сервер, входящий в поставку), то его приложение безо всяких изменений будет работать и в составе большой системы с архитектурой клиент-сервер.

Вот она - масштабируемость на практике - одно и то же приложение можно использовать как для локального, так и для более серьезного клиент-серверного вариантов.

Я уже упомянула о технологиях, которые объединяет **Delphi**. Теперь можно перейти к описанию собственно продукта. Что лежит внутри в коробке, и чем может воспользоваться программист при разработке прикладной системы? Выпущены две версии **Delphi** - одна (**Delphi Client-Server**) адресована для разработчиков приложений в архитектуре “клиент-сервер”, а другая (**Delphi for Windows**) предназначена для остальных программистов. Приложения, разработанные при помощи **Delphi**, можно использовать без выплаты royalty-процентов и без оплаты runtime- лицензий.

Она адресована корпоративным разработчикам, желающим разрабатывать высокопроизводительные приложения для рабочих групп и корпоративного применения.

Клиент-серверная версия включает в себя следующие особенности:

1. SQL Links: специально написанные драйвера для доступа к Oracle, Sybase, Informix, InterBase
2. Локальный сервер InterBase: SQL-сервер для Windows 3.1. СУБД для разработки в корпоративных приложений на компьютере, не подключенном к локальной сети.
3. ReportSmith Client/server Edition: генератор отчетов для SQL-серверов
4. Team Development Support: предоставляет версионный контроль при помощи PVCS компании Intersolve (приобретается отдельно) или при помощи других программных продуктов версионного контроля
5. Visual Query Builder - это средство визуального построения SQL-запросов
6. лицензия на право распространения приложений в архитектуре клиент-сервер, изготовленных при помощи **Delphi**
7. исходные тексты всех визуальных компонент

**Delphi** for Windows представляет из себя подмножество **Delphi** Client-Server и предназначен для разработчиков высокопроизводительных персональных приложений, работающих с локальными СУБД типа dBase и Paradox.**Delphi** Desktop Edition предлагает такую же среду для быстрой разработки и первоклассный компилятор как и клиент-серверная версия (Client/Server Edition). Эта среда позволяет разработчику быстро изготавливать персональные приложения, работающие с персональными СУБД типа dBase и Paradox. **Delphi** позволяет также создавать разработчику DLL, которая может быть вызвана из Paradox, dBase, C++ или каких-нибудь других готовых программ.

1. компилятор Object Pascal (этот язык является расширением языка Borland Pascal 7.0)
2. генератор отчетов ReportSmith 2.5 (у которого, правда, отсутствует возможность работы с SQL-серверами)
3. среда визуального построителя приложений
4. библиотека визуальных компонент
5. Локальный сервер InterBase

В этом обзоре стоит упомянуть еще один продукт, выпущенный компанией Borland для **Delphi**. В RAD Pack for **Delphi** входит набор полезных дополнений, которые помогут разработчику при освоении и использовании **Delphi**. Это учебник по объектному паскалю, интерактивный отладчик самой последней версии, Borland Visual Solutions Pack (набор VBX для реализации редакторов, электронных таблиц, коммуникационные VBX, VBX с деловой графикой и т.п.), Resource WorkShop для работы с ресурсами Borland Pascal 7.0, а также дельфийский эксперт для преобразования ресурсов BP 7.0 в формы **Delphi**.

В первую очередь **Delphi** предназначен для профессионалов-разработчиков корпоративных информационных систем. Может быть, здесь следует пояснить, что конкретно имеется в виду. Не секрет, что некоторые удачные продукты, предназначенные для скоростной разработки приложений (*RAD - rapid application development*) прекрасно работают при изготовлении достаточно простых приложений, однако, разработчик сталкивается с непредвиденными сложностями, когда пытается сделать что-то действительно сложное. Бывает, что в продукте вскрываются присущие ему ограничения только по прошествии некоторого времени.

**Delphi** такие ограничения не присущи. Хорошее доказательство тому - это тот факт, что сам **Delphi** разработан на **Delphi**. Можете делать выводы. Однако **Delphi** предназначен не только для программистов-профессионалов. Я читал в электронной конференции совершенно неожиданные для меня письма, где учителя, врачи, преподаватели ВУЗов, бизнесмены, все те, кто используют компьютер с чисто прикладной целью, рассказывали о том, что приобрели **Delphi for Windows** для того, чтобы быстро решить какие-то свои задачи, не привлекая для этого программистов со стороны. В большинстве случаев им это удается. Поразительный факт - журнал Visual Basic Magazine присудил свою премию **Delphi for Windows**.

Руководители предприятий, планирующие выделение средств на приобретение программных продуктов, должны быть уверены в том, что планируемые инвестиции окупятся. Поэтому одним из оцениваемых факторов должен быть вопрос - а легко ли найти специалиста по **Delphi** и сколько будет стоить его обучение, сколько времени специалист затратит на овладение продуктом. Ответ здесь получить весьма просто - любой программист на паскале способен практически сразу профессионально освоить **Delphi**. Специалисту, ранее использовавшему другие программные продукты, придется труднее, однако самое первое работающее приложение он сможет написать в течение первого же часа работы на **Delphi**. И, конечно же, открытая технология **Delphi** является мощным гарантом того, что инвестици, сделанные в **Delphi**, будут сохранены в течение многих лет.

*Локальный сервер InterBase* - следует заметить, что этот инструмент предназначен только для автономной отладки приложений. В действительности он представляет из себя сокращенный вариант обработчика SQL-запросов InterBase, в который не включены некоторые возможности настоящего сервера InterBase. Отсутствие этих возможностей с лихвой компенсируется преимуществом автономной отладки программ.

*Team Development Support* - средство поддержки разработки проекта в группе. Позволяет существенно облегчить управление крупными проектами. Это сделано в виде возможности подключения такого продукта как Intersolve PVCS 5.1 непосредственно к среде **Delphi**.

*Высокопроизводительный компилятор в машинный код* - в отличие от большинства Паскаль-компиляторов, транслирующих в p-код, в **Delphi** программный текст компилируется непосредственно в машинный код, в результате чего **Delphi**- приложения исполняются в 10-20 раз быстрее (особенно приложения, использующие математические функции). Готовое приложение может быть изготовлено либо в виде исполняемого модуля, либо в виде динамической библиотеки, которую можно использовать в приложениях, написанных на других языках программирования.

Благодаря такой архитектуре приложения, изготовленные при помощи **Delphi**, работают надежно и устойчиво. **Delphi** поддерживает использование уже существующих объектов, включая DLL, написанные на С и С++, OLE сервера, VBX, объекты, созданные при помощи **Delphi**. Из готовых компонент работающие приложения собираются очень быстро. Кроме того, поскольку **Delphi** имеет полностью объектную ориентацию, разработчики могут создавать свои повторно используемые объекты для того, чтобы уменьшить затараты на разработку.

**Delphi** предлагает разработчикам - как в составе команды, так и индивидуальным - открытую архитектуру, позволяющую добавлять компоненты, где бы они ни были изготовлены, и оперировать этими вновь введенными компонентами в визуальном построителе. Разработчики могут добавлять CASE-инструменты, кодовые генераторы, а также авторские help’ы, доступные через меню **Delphi**.

Two-way tools - однозначное соответствие между визуальным проектированием и классическим написанием текста программы Это означает, что разработчик всегда может видеть код, соответствующий тому, что он построил при помощи визуальных инструментов и наоборот.

Визуальный построитель интерфейсов (Visual User-interface builder) дает возможность быстро создавать клиент-серверные приложения визуально, просто выбирая компоненты из соответствующей палитры.

Эта библиотека объектов включает в себя стандартные объекты построения пользовательского интерфейса, объекты управления данными, графические объекты, объекты мультимедиа, диалоги и объекты управления файлами, управление DDE и OLE.

**Delphi** использует структурный объектно-ориентированный язык (Object Pascal), который сочетает с одной стороны выразительную мощь и простоту программирования, характерную для языков 4GL, а с другой стороны эффективность языка 3GL. Программисты немедленно могут начать производить работающие приложения, и им не придется для этого изучать особенности программирования событий в Windows. **Delphi** полностью поддерживает передовые программные концепции включая инкапсуляцию, наследование, полиморфизм и управление событиями.

Это очень важная особенность для разработчиков в среде Windows, поскольку в уже существующие Windows-приложения программист может интегрировать то, что разработает при помощи **Delphi**.

**Задание № 22** – разработать комплекс по учету очередников на получение жилья с отчетом по предприятиям и основаниям включения в очередь.

**Создание базы данных**

Для создания таблицы автономной базы имеется приложение  
*Database Desktop*, которое вызывается командой ***Tools/Database Desktop.***Оно имеет собственное окно, меню и кнопки (заставка этого пункта). Новая таблица создается построителем таблиц, который вызывается командой ***File/New/ТаЫе****.* На запрос тип СУБД выберем Рагаdох 7.

Рассмотрим основные колонки и поля окна конструктора таблиц.  
***File Name -*** идентификаторы полей (не более латинских букв и цифр для совместимости с другими СУБД).

***Туре -*** тип поля (указывается выбором из контекстного меню): *А1рhа* (А - строка из не более чем 255, символов), *Number* (N - число с плавающей точкой), *Моnеу* ($ - аналогичен типу *NumЬег*, но добавляется денежный знак), *Short* (S - целое число в диапазоне -32768..32767), *LongIntereger* (I целое число в диапазоне-2147483.. 2147483647), ВСD (# - число в двоично-десятичном формате), *Datе* *(D -* дата). *Time* (Т - время), *Timestamp* (@ - дата и время), *Меmо* (М - строка любой длины), *Formatted Memo* (F - аналогичен *Меmо*, но может содержать форматированный текст), *Graphic* (G -изображение ВМР, РСХ, ТIF, СИР, ЕРS), ОLЕ (О - объект ОLЕ), *Logical* (L логическое - Т, F), *Autoincrement* (+ - при добавлении записи, автоматически формируется уникальное значение), *Вinary* (В – последовательность байтов любой длины), *Вуtes* (У последовательность не более, чем 255 байтов).

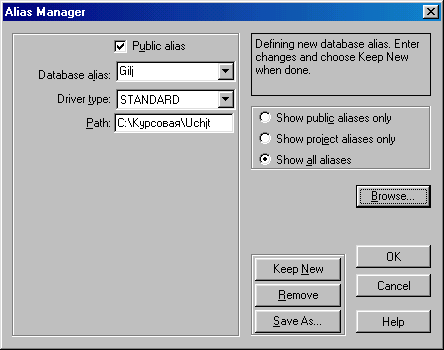
***Size-*** размер поля в байтах.

***Кеу -*** признак первичного ключевого поля (двойной щелчок по колонке).  
***Таbles properties -*** дополнительные свойства таблицы*: Secondary Index* (задание вторичных индексов), *Validity Checks* (ограничения на ввод значений полей). *Рassword Security* (определение пароля), *Referential Intergity* (определение ссылочной целостности между таблицами), *ТаЫе Language*(задание языка), *ТаЫе Lоокор* (задание полей просмотра), *Depending ТаЫеs*(зависимые дочерние таблицы).

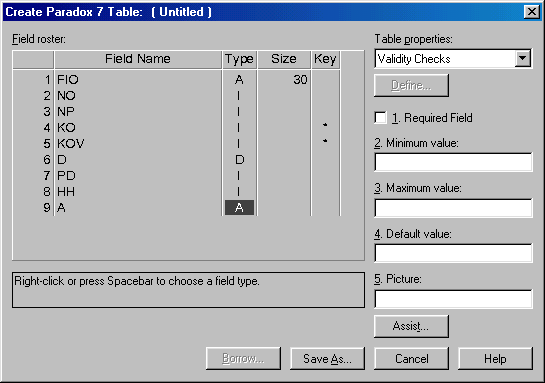
*Requierd Filed -* признак обязательного заполнения поля.  
*Мiniтит..., Махiпит... -* начало и конец диапазона допустимых значений.  
*Default vа1ие -* значение по умолчанию при добавлении пустой записи.  
*Р1сture,Assist -* шаблон и построитель шаблона значения поля.

***Вторичные ключи (индексы)*** создаются путем выбора из списка свойств таблицы строки ***Secondary Indexes,*** и после нажатия кнопки ***Define*** выводится окно ***Define Secondary Indexes***. Используя кнопки со стрелками или двойными щелчками, перенесем нужное поле или несколько полей, если ключ составной, в список индексов. Имеются переключатели: ***Unique*** (уникальный индекс), ***Саse sensitive*** (учет регистров строковых полей), ***Descending*** (сортировка по возрастанию). Кнопкой ***ОК.*** производится выход с указанием имени индекса. Поле, которое является первичным ключом другой родительской таблицы, называют внешним ключом. Для связи таблиц обычно создают индексы по внешним ключам. После формирования таблицы она сохраняется, если нажать кнопку *Save As****,*** под указанным именем; при этом в поле имени файла можно указать как  
полное имя с каталогами, так и псевдоним базы (Аlias). Псевдоним позволяет не привязываться к каталогу, который может меняться в дальнейшем.

Для примера создадим базу из четырех таблиц: Очередники, Организации, Города и Основания. В начале создадим каталог Жилье, в котором будет находится наша база. Далее присвоим создаваемой базе псевдоним Учет.  
Длясоздания псевдонима выполним команду ***Тоо1s/Аlias Мапаger,*** и появится окно ***Аlias Маnаgег***.



Нажмем кнопку ***New*** и в поле ***Database аlias*** введем имя псевдонима Учет, кнопкой ***Вrowse*** выберем наш каталог с базой Жилье, далее ***Кеер* New** и ***ОК.***



Командой ***File/Working Directory*** установим текущий каталог,  
выбрав псевдоним Учет в списке ***Аliases****.* Займемся созданием таблиц.

Таблица Очередники будет иметь поля: ФИО, № очереди, № приказа, Код организации, Код основания, Дата, Паспортные данные, Число членов семьи, Адрес. Первичными ключами служат поля Код организации и Код основания. Все поля, являются обязательными.

Таблицы Организации и Основания будут иметь ключевые поля (код организации\оснолвания).

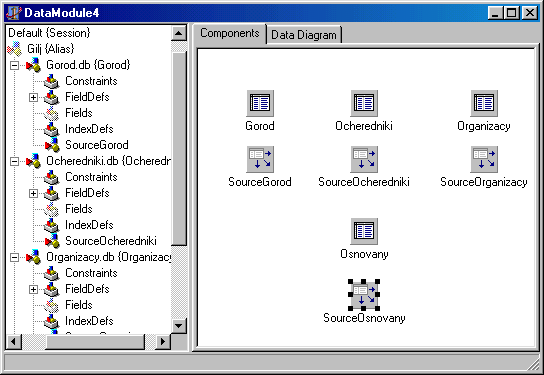
Для корректировки и просмотра открытой таблицы имеются команды ***ТаЫе/Restructure*** (кнопка *Restructure*) и ***ТаЫе/Into Structure****.* Для переименования таблицы следует сохранить ее под новым именем. Для заполнения и редактирования таблицы, после ее связывания с другими таблицами, испочьзуется кнопка *Edit Data* и меню *Record* приложения Database Deskot (кириллицу при вводе использовать нельзя).

**Создание модуля данных**

Модуль данных содержит объекты, связанные с базой, и использует­ся для централизованного управления этими объектами. Его создание производится командой ***File/New/Data Module***. Имя созданного модуля данных лучше сохранить для наглядности под новым именем, на­пример: DataModulGilj. Разместим компонент ***Table*** *с* панели Data Ac­cess на странице Components и выделим его. Зададим свойства:

***DataBaseNaine****=Ychjt* (псевдоним базы), ***TableName****=Oheredniki* (имя табли­цы), **Name**=Oheredniki (имя компонента приравняем имени таблицы для наг­лядности

***Active****=True* (чтобы можно было обращаться к таблице из прог­раммы). Аналогично сделаем для трех оставшихся таблиц. Далее раз­местим компонент ***Data Source*** (источ­ник данных) и выделим его. Зада­дим свойства: ***DataSet=Ocheredniki*** (имя таблицы) , ***Name****=SourceOcheredniki* (имя компонента).



То же проделаем и с другими таблицами. Получим модуль. Источник данных служит посредником между таблицей базы и экранными формами, что позволяет переходить на другие СУБД, меняя только настройку компонента - источника данных, а не программы. После открытия проекта нужно вызвать этот модуль командой ***View/Units/Unit2.***

Поля в таблице можно использовать как динамические (dynamic) либо как постоянные (persistent). ***Динамические поля*** должны быть в таблице, и их можно редактировать. ***Постоянные поля*** (виртуальные или псевдополя) могут отсутствовать в таблице, и их значения вычисляются. К имени постоянного поля автоматически добавляется имя компонента Table.

При ссылках на объекты модуля данных объект уточняется именем модуля DataModule2 (DataModule2.SourceOcheredniki).

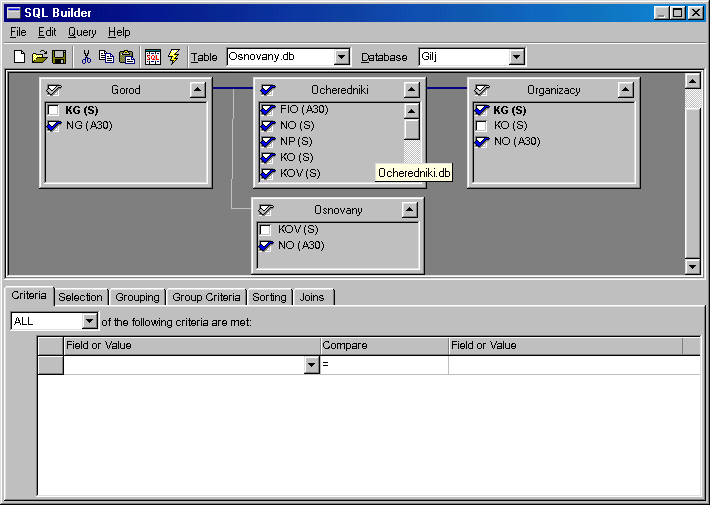
***Рекомендуется*** для всех полей задавать словесные наименования полей в их свойствах Display Label через редактор полей, которые выводятся в формах и компонентах вместо латинских имен полей.

***Рекомендуется*** модуль базы данных создавать под содержательным именем в папке с базой, включать его в новый проект командой ***Рroject/Add to project*** и подключать его к головному или другому модулю командой ***File/Use Unit.***

**Создание запросов**

Запросы служат для выборки нужных записей и объединения нескольких таблиц в одну. С помощью запросов реализуется реляционный доступ к базе данных, который автоматизирует и унифицирует процедуры доступа к распределенным многопользовательским базам. Для создания запросов используются команды языка структурированных запросов (SQL). Этот язык стандартизован и используется в различных СУБД, что унифицирует доступ к данным В Delphi имеются визуальные средства по формированию команд SQL, это упрощает и ускоряет создание простых запросов и не требует знания деталей использования команд SQL.

Для примера составим запрос, в который включим все поля таблицы очередников и городов. Разместим компоненты ***Querу*** (со свойсгвом Асtivе=Тruе), ***DataSource*** (со свойствами Namе=SourceBazoviZapros, DataSet=Quегу1) с панели Data Access на странице Соmponents модуля данных форме и выберем команду ***SQL Builder*** (построитель запросов), и появится окно построителя запросов



В списке ***Database*** выберем базу Uchjt Используя список ***Таblе****,* выберем таблицы очередники, города, организации, основания. Мышкой установим связь между этими таблицами по полям NO и КO соответственно Галочкой пометим поля, выводимые в запрос На странице ***Criteria*** зададим условия отбора записей.

На странице ***Sorting*** задается список полей сортировки в порядке убывания приоритетов Порядок сортировки задается кнопками ***А..Z*** (по возрастанию), ***Z...А*** (по убыванию)

Для проверки запроса нажмем кнопку ***Execute Querу.*** Кнопкой ***Savе the current query*** сохраним его в базе под именем ВаzoviZapros, ею можно посмотреть и откорректировав редактором запроса, вызываемого кнопкой ***Show and Edit SQL***.

Для отображения запроса на форме, в редакторе полей дпя объекта Оuегу1 добавим все или отдельные поля запроса. Разместим компонент DBGrid на форме со значением его свойства DataSource, равным DataModule2. SourceBazoviiZapros, сформируем столбцы этой таблицы

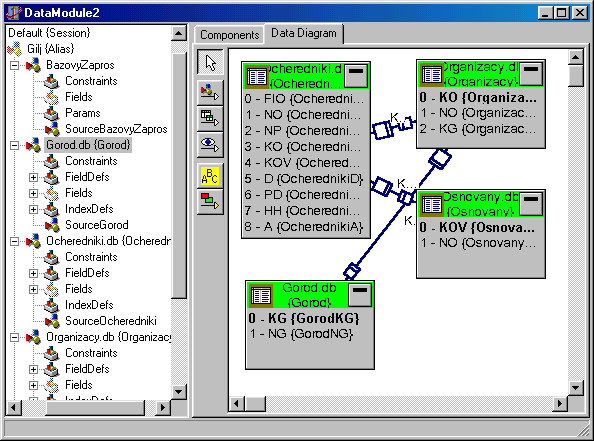
Запрос может быть выполнен непосредственно из программы  
методами ЕхесSQL, Ехесute или из проводника SQL Ехр1огег, вызываемого командой ***DataBase/Explore****.* В проводнике можно набрать текст запроса на странице ***Епter SQL****,* выполнить его кнопкой ***Ехcute Querу*** и сохрани гь его в текстовом файле командой ***Оbject/Save As.***

Установка связей между таблицами

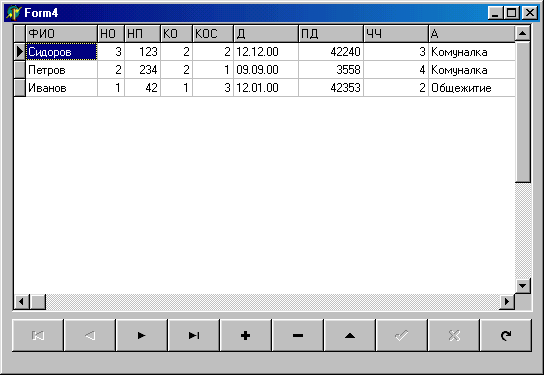
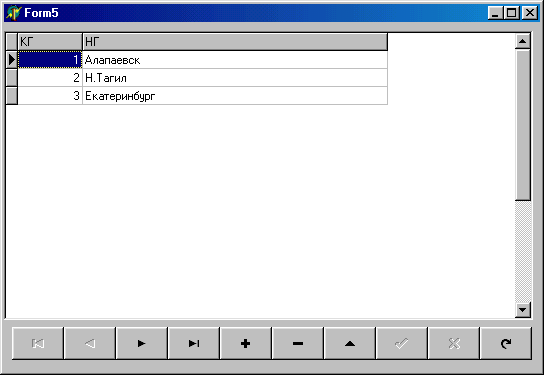
Имя источника главной (родительской) таблицы, с которой останавливается связь, и имя поля главной таблицы, по которому связывается таблица, задаются в свойствах подчиненной (дочерней) таблицы МаsterSouгсе и МаsterFields соответственно В нашем примере цля таблицы организаций этим свойствам присвоим значения МаsterSouгсе =SourceOcheredniki, Маster,  
МаsterFields =KO

Существует визуальный способ установки связей Выберем модуль данных и откроем страницу Data Diagram окна DataModule2 Перенесем последовательно мышкой из проводника этого окна узлы с именами таблиц Ocheredniki, Orgahizacy, Gorod, Osnovany на страницу Data Diagram Щелкните на кнопке ***Маster Datail*** Проведите линию при нажатой левой кнопке мыши от таблицы Ocheredniki до таблицы Organizacy; появится окно Field Link Designer. Установите значения для полей и нажмите кнопку ***Аdd*** и ***ОК,*** связь установлена, и появится линия связи. Аналогично устанавливается связь и остальных таблиц. Объекты схемы данных можно прокомментировать: нажмите кнопку ***Соттеnt В1осК*** и щелкните мышкой на свободном месте схеме и в появившемся прямоугольнике введите текст комментария, щелкните по кнопке ***Соттеnt Аlludе*** и протяните линию от комментария к нужному объекту.

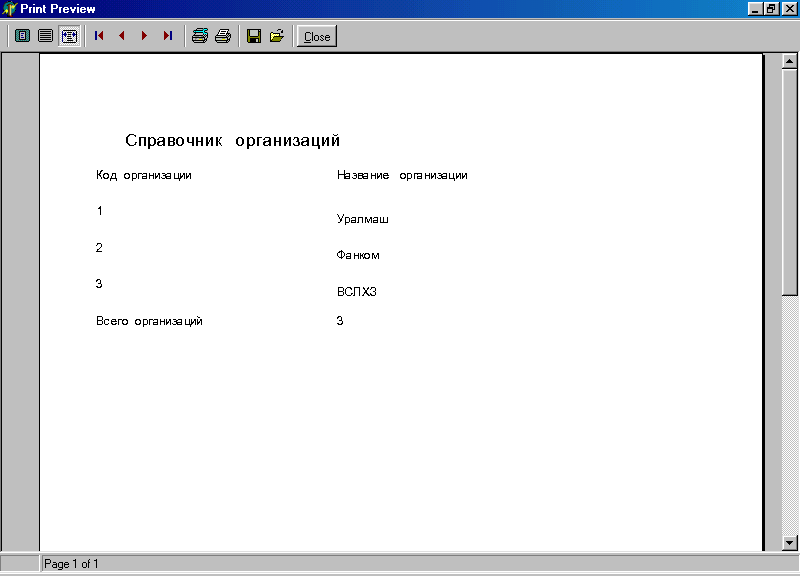
Мастер форм



Мастер форм вызывается командой **DataBase/Form Wizard*.*** На первом шаге выбирается тип формы: ***Сreate a simple form*** (простая форма), ***Create a master/detail form*** (составная форма, состоящая из главной и детальной форм, например: очередники и организации). Выбирается тип источника данных (таблицы или запросы), выбираются таблицы (запросы) и их поля, выводимые в форму. Если форма составная, то указываются поля связи главной и детальной таблицы (в нашем примере - код организации).Выбирается тип формы: вертикальная, горизонтальная или табличная.  
Полученную формуможно откорректировать.

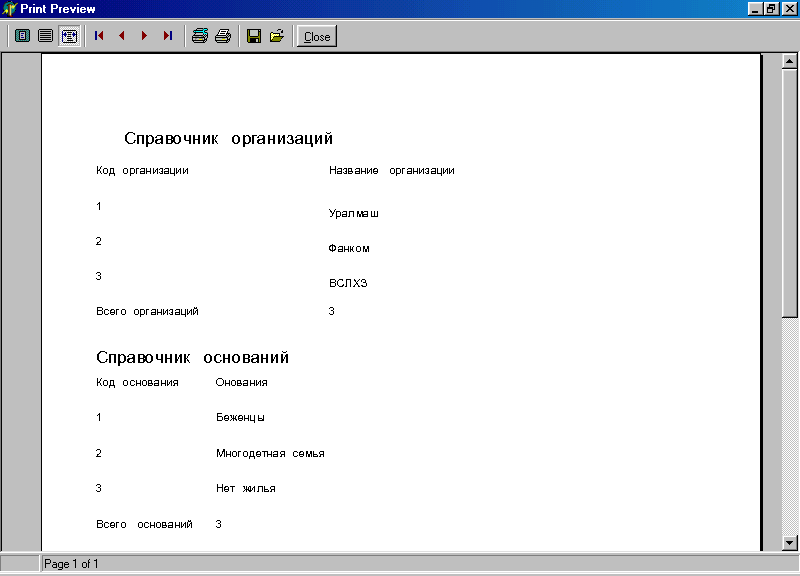


**Отчеты**



Выделяются четыре типа отчетов: простой,

с группировкой данных, связанный отношением "мастер/детальный" и составной, объединяющий несколько отчетов.



Компоненты отчета находятся на панели QReport. Создание отчета обычно начинается с размещения на отдельной форме компонента QuickRep. Он выполняет роль формы, на которой размещаются другие компоненты отчета.

Отчет делится на горизонтальные полосы (разделы). Каждая полоса является отдельным настраиваемым компонентом, тип ее задается в свойствах Ваnds, ВаndТуре компонентов QuickRep, QRBand соответственно. При проектировании отчета можно его просмотреть командой Ргеview контекстного меню. После формирования отчета он просматривается в окне предварительного просмотра, вызываемого методом Ргеview, или выводится на печать методом Ргint. Можно создать отдельное окно предварительного просмотра, используя компонент QRPreview. Рекомендуется для отчета выделять отдельную невидимую форму.

Пользователь может воспользоваться готовыми заготовками (шаблонами) отчетов, если выполнит команды ***File/New/Forms****.* Имеются три шаблона: Quick Report Master/Detail (отчет из главной и подчиненной таблицы), Quick Report Labels (почтовые этикетки),Quick Report List (опись). Можно воспользоваться мастером отчета, который вызывается командой ***File/New /Виziness/* Quick Report *Wizard****.* Используемые заготовки или отчет, полученный мастером, можно откорректировать.

Сочетание возможностей быстрого прототипирования приложений с технологиями уровня предприятия обеспечивает плавное и предсказуемое развитие проектов любого масштаба. Следование стандартам индустрии и открытость к взаимодействию с любыми частными решениями гарантирует успех проектов, разрабатываемых с использованием Delphi.