УДК 37.022:681.3 (575.2)

Т.Н. РУБЛЕВА

E-mail: ksucta@elcat.kg.

МЕТОДИКА ПРЕПОДАВАНИЯ БАЗ ДАННЫХ В СРЕДЕ DELPHI-7

Бул макалада Delphi-7 программасын окутуунун усулдары каралган. Базаны түзүүнүн жана иштеп чыгуунун этаптары сунуш кылынган.

В данной статье рассмотрена методика преподавания баз данных в приложении Delphi-7. Предложены этапы разработки и создания базы данных.

In this article consideration a method of teaching data base in Delphi-7 application. It has stage of creation of data base.

В настоящее время, когда происходит процесс информатизации общества, главным объектом становится информация. Для решения разнообразных практических задач требуется обработка больших массивов данных об объектах реального мира, требующая умений организации хранения, поиска, сортировки, классификации, систематизации информационных ресурсов. Поэтому для более успешной и эффективной деятельности студенты должны знать основные принципы работы систем управления базами данных (СУБД), а также уметь осмысленно оперировать основными объектами баз данных.

Возможность создавать программы, обслуживающие базы данных, – важная отличительная особенность Delphi. Можно сказать, что Delphi − это функционально полная реляционная СУБД. В ней предусмотрены все необходимые средства для определения и обработки данных, а также для управления ими при работе с большими объемами информации. Кроме того, Delphi дает максимальную свободу в задании типа данных. Поскольку Delphi является современным приложением Windows, то это приложение позволяет использовать все возможности BDE (динамического обмена данными) и OLE (связи и внедрения объектов), т.е. осуществлять обмен данными между Delphi и любым другим поддерживающим DDE приложением Windows.

Мощность и гибкость Delphi при работе с базами данных основаны на низкоуровневом ядре – процессоре баз данных Borland Database Engine (BDE). BDE – машина баз данных фирмы Borland (набор библиотек), выполняющая действия по доступу к данным и проверке их правильности. BDE «умеет» работать с таблицами самых распространенных СУБД, такими как файл-серверных (dBase, Paradox, FoxPro) и клиент-серверных (InterBase, MS SQL Server, Oracle и др.). В BDE имеется собственный интерпретатор языка SQL,что позволяет создавать запросы не только к серверам БД, но и к таблицам файл-сервера /1/.

База данных – это информационная модель, позволяющая в упорядоченном виде хранить данные о группе объектов, обладающих одинаковым набором свойств.

В процессе развития теории и практического использования баз данных, а также средств вычислительной техники создавались СУБД, поддерживающие различные даталогические модели: иерархические, сетевые и реляционные.

Иерархические БД состоят из упорядоченного набора деревьев; более точно, из упорядоченного набора нескольких экземпляров одного типа дерева. Тип дерева состоит из одного «корневого» типа записи и упорядоченного набора из нуля или более типов поддеревьев (каждый из которых является некоторым типом дерева). Тип дерева в целом представляет собой иерархически организованный набор типов записи. Этот тип связи можно отобразить следующей схемой: предок – потомок, между которыми поддерживается связь. Никакой потомок не может существовать без своего родителя, причем предок должен быть один.

Простота организации, наличие заранее заданных связей между сущностями, сходство с физическими моделями данных позволяли добиваться приемлемой производительности иерархических СУБД на медленных ЭВМ с весьма ограниченными объемами памяти.

Сетевые модели также создавались для малоресурсных ЭВМ. Сетевой подход к организации данных является расширением иерархического. В этой модели потомок может иметь любое число предков. Сетевая БД состоит из набора записей и набора связей между этими записями, а если говорить более точно, из наборов экземпляров каждого типа из заданного в схеме БД набора типов записи и набора экземпляров каждого типа из заданного набора типов связи. Тип связи определяется для двух типов записи: предка и потомка.

Сложность практического использования иерархических и сетевых СУБД заставляла искать иные способы представления данных.

Сегодня наиболее распространены реляционные (основанные на двумерных таблицах) модели данных. Любая система данных может быть сведена к набору таблиц: строки обычно называют записями, а столбцы – полями. Реляционная база данных представляет собой множество взаимосвязанных таблиц, каждая из которых содержит информацию об объектах определенного типа.

Технология создания баз данных

Каждая информационная система в зависимости от ее назначения имеет дело с частью реального мира, которую принято называть предметной областью (ПО) системы. ПО может относиться к любому типу организации: банку, университету, заводу, магазину и т.д.

Предметная область информационной системы – это совокупность реальных объектов (сущностей), которые представляют интерес для пользователей.

Объект (сущность) – предмет, процесс или явление, о котором собирается информация, необходимая для решения задачи. Объектом может быть человек, предмет, событие.

Каждый объект характеризуется рядом основных свойств – атрибутов. Атрибутом называется поименованная характеристика объекта. Атрибут показывает, какая информация

должна быть собрана об объекте. Например, объект – клиент банка; атрибуты – номер счета, адрес, сумма вклада.

Этапы создания базы данных:

1. Анализ предметной области, который заканчивается построением информационной структуры (концептуальной схемы). От того, насколько грамотно спроектирована БД, зависят эффективность будущей СУБД и ее полезность для пользователя.

2. Далее выполняется выбор информационных объектов, задание необходимых свойств для каждого объекта, выявление связей между объектами, определение ограничений, накладываемых на информационные объекты, определение типов связей между ними.

Все информационные объекты предметной области связаны между собой. Соответствия, отношения, возникающие между объектами предметной области, называются связями. Связанные отношениями таблицы взаимодействуют по принципу главная (master) – детальная (detail). Для установки связи между таблицами используется свойство MasterFields подчиненной таблицы, где указываются связываемые поля в родительской и дочерней таблицах.

Различаются связи нескольких типов, для которых введены следующие обозначения:
а) один к одному (1:1); один ко многим (1:M); многие ко многим (M:M).

Связь один к одному предполагает, что в каждый момент времени одному экземпляру информационного объекта A соответствует не более одного экземпляра информационного объекта B и наоборот.

При связи один ко многим одному экземпляру информационного объекта A соответствует 0, 1 или более экземпляров объекта B, но каждый экземпляр объекта B связан не более чем с 1 экземпляром объекта A.

Связь многие ко многим предполагает, что в каждый момент времени одному экземпляру информационного объекта A соответствует 0, 1 или более экземпляров объекта B и наоборот.

3. Создание псевдонима БД. В Delphi проблема передачи в программу информации о месте нахождения файлов БД решается путем использования псевдонима. Псевдоним (alias) – это краткое имя, поставленное в соответствие реальному полному имени каталога БД, то есть это путь доступа к таблицам базы данных. Он сохраняется в отдельном конфигурационном файле в произвольном месте на диске. Такой подход дает возможность располагать данные в любом месте, не перекомпилируя при этом программу. Кроме пути доступа, в алиасе указываются тип базы данных, языковый драйвер и много другой управляющей информации /1/.

Замечание. Для каждого нового разрабатываемого проекта необходимо создавать свою папку для хранения разрабатываемых таблиц БД и самого проекта.

4. Разработка структуры таблиц БД и создание этих таблиц с помощью Database Desktop. Database Desktop − это утилита, которая поставляется вместе с Delphi для интерактивной работы с таблицами различных форматов локальных баз данных – Paradox и dBase /2/. Для полей таблиц БД можно выбрать следующие типы: Number, Short, LongInteger (числовое поле), Money

(денежное), Date (дата), Memo (поле примечания), Graphic (графическое), Logical (логическое), Autoincrement (поле счетчика) /2/.

5. После создания таблицы с ней можно связать некоторые свойства:

− *Validity Checks* (проверка правильности) − относится к полю записи и определяет минимальное и максимальное значения, а также значение по умолчанию. Кроме того, позволяет задать маску ввода.

− *Table Lookup* (таблица для «подсматривания») − позволяет вводить значение в таблицу, используя уже существующее значение в другой таблице.

− *Secondary Indexes* (вторичные индексы) − позволяют обращаться к данным в порядке, отличном от порядка, задаваемого первичным ключом.

− *Password Security* (парольная защита) − позволяет закрыть таблицу паролем.

− *Table Language* (язык таблицы) − позволяет задать для таблицы языковый драйвер.

6. Создание форм. Имеется несколько основных компонент (объектов), которые можно использовать для доступа к БД и отображения данных. Эти объекты могут быть разделены на три группы:

- Невизуальные компоненты: наборы данных (Data Set), непосредственно связывающиеся с базой данных. Для BDE это такие компоненты, как таблица (TTable), запрос (TQuery). Эти компоненты размещены на странице BDE и используются для управления таблицами и запросами.

Компоненты визуализации данных и управления данными: сетка (TDBGrid) отображает содержимое НД, в которой столбцы соответствуют полям набора данных, а строки – записям; однострочное редактируемое поле (TDBEdit) используется для отображения записей набора данных; DBImage − этот компонент прозволяет отображать одно из трех поддерживаемых Delphi изображений: растрового рисунка, значка или мета-файла); комбинированный список (DBComboBox) применяется для выбора нужного значения из списка; DBNavigator предоставляет пользователю удобное средство для перемещения по записям набора данных, позволяет выполнять вставку, удаление и редактирования записей; и другие компоненты страницы Data Control. Эти компоненты показывают данные пользователю и позволяют ему просматривать и модифицировать их /2/.

- Компонент − источник данных (TDataSource) расположен на странице Data Access,

осуществляет обмен информацией между компонентами первого типа и компонентами визуализации и управления данными. Связь этих компонентов друг с другом и с базой данных можно представить схемой, приведенной на рис.1.

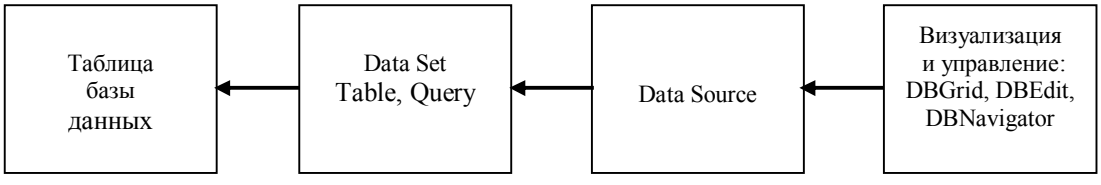

Рис.1. Связь между визуальными, невизуальными и связующими компонентами.

7. Создание объектов-полей для доступа к данным. Создание объектов-полей можно считать признаком хорошего стиля программирования, так как они упрощают доступ к данным и дают программисту дополнительные возможности /2/.

С помощью редактора полей можно создать новые поля одного из трех типов:

- Data − поля содержат произвольные данные.
- Calculate − вычисляемые поля.
- Lookup − подстановочные поля.

Поле первого типа будет отображаться в сетке пустой колонкой, которую можно заполнить в обработчике события OnGetText этого поля.

Создание вычисляемых полей − одно из наиболее ценных свойств редактора DataSet. Вычисляемое поле предназначено для отображения данных, которые вычисляются в ходе выполнения программы обычно с помощью значений из других полей той же записи. Такое поле заполняется в обработчике события OnCalcFields набора данных.

Для заполнения подстановочного поля данные берутся из нужного поля другого набора данных (НД). Непременным условием создания подстановочного поля является требование реляционной связи главного НД с таблицей подстановки. В главной таблице должен быть создан первичный ключ.

8. Выполнение сортировки, фильтрации и поиска данных.

Сортировка записей выполняется автоматически по текущему индексу, при смене индекса происходит автоматическое переупорядочивание записей, таким образом, сортировка возможна для полей, для которых создан индекс. Направление сортировки определяет параметр текущего индекса ixDescending, по умолчанию он выключен, и сортировка выполняется по возрастанию. Задать текущий индекс можно, использовав свойство IndexName или IndexFieldName, так как главный индекс (ключ) не имеет имени, то сортировка выполняется с помощью свойства IndexFieldName /2/.

Для поиска записей в таблице используются методы Locate и Lookup. Метод Locate ищет первую запись, удовлетворяющую критерию поиска, и если такая запись найдена, делает ее текущей. Метод Lookup находит запись, удовлетворяющую условию поиска, но не делает ее текущей, а возвращает значения некоторых ее полей.

Поиск записей по индексированным полям. Используются следующие методы: FindKey, SetKey, EditKey, GotoKey.

Фильтрация записей. Фильтр позволяет отображать данные согласно заданному критерию отбора. Для ограничения записей используются два фильтра: фильтр по выражению и фильтр по диапазону /3/.

Фильтрация по выражению ограничивает набор данных записями, которые удовлетворяют выражениям, записанным в фильтре. Этот вид фильтрации применим к любым полям, в том числе и неиндексированным.

Фильтр по диапазону позволяет отобразить записи, значения которых попадают в заданный диапазон. Такой вид фильтрации проходит только по индексированным полям, что значительно ускоряет процесс.

9. Создание запросов с помощью языка SQL. Язык SQL (Structured Query Language) − язык структурированных запросов предназначен для манипулирования данными в реляционных базах данных и состоит из 4-х основных команд: SELECT (выбрать); INSERT (вставить); UPDATE (обновить); DELETE (удалить).

SQL можно использовать для создания и просмотра таблиц, формирования состава полей НД, отбора записей по сложным критериям, объединения таблиц, создания между таблицами требуемых отношений /2/.

10. Доступ к полям. К значению поля можно обратиться при помощи свойств Value и AsXXXX, например: Table1Zena.Value:=100.93; Table1.FieldByName ('Zena').AsFloat:=1000.2.

11. Построение диаграмм. Delphi позволяет строить диаграммы на основе данных, вводимых пользователем или генерируемых программно. Для построения диаграмм используется компонент DBChart, расположенный на вкладке Data Control палитры компонент. Компонент DBChart предназначен для отображения различных типов диаграмм на основе данных выбранной таблицы БД. Пользователь может использовать 11 типов стандартных и 7 типов нестандартных диаграмм. В качестве источника данных для построения диаграммы указывается таблица набора данных через свойство DataSet. Установка требуемых свойств диаграмм выполняется в редакторе диаграмм, для вывода которого необходимо выполнить двойной щелчок на компоненте DBChart.

12. Создание отчета. Важной составной частью при разработке БД является вывод данных на печать и получение отчета.

Для создания отчета могут быть использованы компоненты страницы QReport. На странице QReport размещено более 20 компонент, с помощью которых программист может создавать довольно сложные отчеты. Для создания отчета используется отдельная форма, на которую устанавливается компонент QuickRep. Этот компонент является основным контейнером для страницы отчета и может, в свою очередь, содержать другие компоненты. Отчет в основном строится из компонентов-полос QRBand, с помощью которых формируются общий заголовок, заголовки колонок отчета, область для отображения собственно данных из таблиц БД. Однако надо заметить, что компоненты QReport имеют один существенный недостаток: их нельзя использовать в кроссплатформенных приложениях /3/.

В связи с этим в пакет Delphi-7 входит средство для генерации и печати отчетов – RaveReport. Компоненты вкладки Rave обладают большими возможностями и, главное, поддерживают кроссплатформенные приложения. Отчет создается специальной машиной генерации отчета по указаниям, получаемым из файла проекта отчета. Файл проекта

разрабатывается с помощью утилиты Rave Reports Designer. Компоненты, обеспечивающие связь файла проекта с реальными данными из БД, расположены на странице Rave.

Предлагаемая технология создания базы данных позволит студентам разработать профессиональную базу данных, используя широкие возможности приложения Delphi.

СПИСОК ЛИТЕРАТУРЫ

1. Фаронов В.В. Система программирования Delphi-7: Наиболее полное руководство. − СПб.: Питер, 2003. − 888 с.

2. Фаронов В.В. Программирование баз данных в Delphi-6: Учебный курс. − СПб.: Питер, 2003. − 464 с.

Пономарев В.С. Базы данных в Delphi 7: Самоучитель. − СПб.: Питер, 2003. − 224с.