Мы познакомились с работой Excel и знаем, что это приложение создано специально для решения задач обработки табличных данных.

Существуют системы (приложения) для решения иных классов задач. В частности, очень большую роль играют сейчас программы (приложения, системы), цепь которых – хранение данных и выдача данных по запросу пользователя.

Использование ЭВМ именно для решения этого класса задач становится всё более массовым явлением.

Смело можно сказать, что такие задачи и необходимость их решения существуют в любой фирме, на любом предприятии.

Основное понятие для подобного круга задач – база данных. Так называется файл или группа файлов стандартной структуры, служащая для хранения данных.

Для разработки программ, систем программ, работающих с базами данных, используются специальные средства – системы управления базами данных (СУБД).

СУБД включает, как правило, специальный язык программирования и все прочие средства, необходимые для разработки указанных программ.

В настоящее время наиболее известными СУБД являются FOXPRO и ACCESS. Последняя входит в состав профессионального пакета MS Office 97.

Это современные системы с большими возможностями, предназначенные для разработки сложных программных комплексов, и знакомство с ними для пользователя ЭВМ исключительно полезно, но в рамках настоящего пособия осуществить его затруднительно.

Понятие базы данных

База данных (БД) – это совокупность массивов и файлов данных, организованная по определённым правилам, предусматривающим стандартные принципы описания, хранения и обработки данных независимо от их вида.

База данных (БД) – совокупность организованной информации, относящейся к определённой предметной области, предназначенная для длительного хранения во внешней памяти компьютера и постоянного применения.

Виды БД:

1.*Фактографическая* – содержит краткую информацию об объектах некоторой системы в строго фиксированном формате;

2.*Документальная* – содержит документы самого разного типа: текстовые, графические, звуковые, мультимедийные;

3.*Распределённая* – база данных, разные части которой хранятся на различных компьютерах, объединённых в сеть;

4.*Централизованная* – база данных, хранящихся на одном компьютере;

5.*Реляционная* – база данных с табличной организацией данных.

Одно из основных свойств БД – независимость данных от программы, использующих эти данные.

Работа с базой данных требует решения различных задач, основные из них следующие:

Создание базы, запись данных в базу, корректировка данных, выборка данных из базы по запросам пользователя.

Задачи этого списка называются стандартными.

Следующее понятие, связанное с базой данных: программа для работы с базой данных – это программа, которая обеспечивает решение требуемого комплекса задач. Любая подобная программа должна уметь решать все задачи стандартного набора.

База данных в разных системах имеет различную структуру.

В ПВЭМ обычно используются реляционные БД – в таких базах файл является по структуре таблицей. В ней столбцы называются полями, строки – записями.

В БД содержатся банные некоторого множества объктов. Каждая запись содержит данные одного объекта. Каждая такая БД определяется именем файла, списком

полей, шириной полей. Например, БД Школа (Ученик, Класс, Адрес).

Примером БД может служить расписание движения поездов или автобусов. Здесь каждая строчка – запись отражает данные строго одного объекта. База включает поля: номер рейса, маршрута следования, время отправления и т.д.

Классическим примером БД является и телефонный справочник. Запрос к базе данных – это предписание, указывающее, какие данные пользователь желает получить из базы.

Некоторые запросы могут представлять собой серьёзную задачу, для решения которой потребляется составлять сложную программу. Например, запрос к базе – автобусному расписанию: определить разницу в среднем интервале отправления автобусов из Ростова в Таганрог и из Ростова в Шахты.

Объекты для работы с базами данных

Для создания приложения, позволяющего просматривать и редактировать базы данных, нам потребуется три звена:

• *набор данных*

• *источник данных*

• *визуальные элементы управления*

В нашем случае эта триада реализуется в виде:

• *Table*

• *DataSource*

• *DBGrid*

Table подключается непосредственно к таблице в базе данных. Для этого нужно установить псевдоним базы в свойстве DataBaseName и имя таблицы в свойстве TableName, а затем активизировать связь: свойство Active = true.

Однако, поскольку Table является невизуальным компонентом, хотя связь с базой и установлена, пользователь не в состоянии увидеть какие – либо данные. Поэтому необходимо добавить визуальные компоненты, отображающие эти данные. В нашем случае это сетка DBGrid. Сетка сама по себе «не знает», какие данные ей

нужно отображать, её нужно подключить к Table, что и делается через компонент – посредник DataSource.

А зачем нужен компонент – посредник? Почему бы сразу не подключаться к Table?

Допустим, несколько визуальных компонентов – таблица, поля ввода и т.п. подключены к таблице. А нам нужно быстро переключить их все на другую подобную таблицу. С DataSource это сделать несложно - достаточно просто поменять свойство DataSet, а вот без DataSource пришлось бы менять указатели у каждого компонента.

Приложения баз данных – нить, связывающая БД и пользователя:

БД – набор данных – источник данных – визуальные компоненты – пользователь

Набор данных:

• Table (таблица, навигационный доступ)

• Query (запрос, реляционный доступ)

Визуальные компоненты:

• Сетки DBGrid , DBCtrlGrid

• Навигатор DBNavigator

• Всяческие аналоги Lable , Edit и т.д.

• Компоненты подстановки

Типы данных в базах

В Access можно определить следующие типы полей:

• *Текстовый* – текстовая строка; максимальная длина задаётся параметром «размер», но не может быть больше 255

• *Поле MEMO* – текст длиной до 65535 символов

• *Числовой* – в параметре «Размер поля» можно задать поле: байт, целое, дейсвительное и т.п.

• *Дата/время* – поле, хранящее данные о времени.

• *Денежный* – специальный формат для финансовых нужд, по сути являющийся числовым

• *Счётчик* – автоинкрементное поле. При добавлении новой записи внутренний счётчик таблицы увеличивается на единицу и записывается в данное поле новой записи. Таким образом, значения этого поля гарантированно различны для разных записей. Тип предназначен для ключевого поля

• *Логический* – да или нет, правда или ложь, включен или выключен

• *Объект OLE* – в этом поле могут храниться документы, картинки, звуки и т.п. Поле является частным случаем BLOB – полей (Binary Large Object), встречающихся в различных базах данных

• *Гиперссылка* – используется для хранения ссылок на ресурсы Интернета. Встречается не во всех форматах баз данных. К примеру, такого типа нет в dBase и Paradox

• *Подстановка*

Типы данных в таблицах Access :

• *Текстовый*

• *Поле MEMO*

• *Числовой*

• *Дата\время*

• *Денежный*

• *Счётчик*

• *Логический*

• *Объект OLE*

• *Гиперссылка*

Не надо забывать про индексы.

Связывать таблицы.

Связь с обеспечением целостности контролирует каскадное удаление и модификацию данных.

Монопольный доступ к БД нужен для того, чтобы производить в ней фундаментальные изменения.

Основные понятия и элементы баз данных

Базы данных понадобились тогда, когда возникла потребность хранить большие объёмы однотипной информации, уметь её оперативно использовать. Базами данных (в широком понимании этого слова) пользовались на протяжении всей истории жрецы, чиновники, купцы, ростовщики, алхимики.

Основное требование к базам данных – удобство доступа к данным, возможность оперативно получить исчерпывающую информацию по любому интересующему вопросу (важно не только то, что информация содержится в базе, важно то, насколько она хорошо структирована и целостна).

Лишь только появились и распространились компьютеры, почти сразу на них возложили тяжёлый и кропотливый труд по обработке и структурированию данных, появились базы данных (БД) в их нынешнем понимании.

Согласно современным требованиям к базам данных, информация, содержащаяся в них, должна быть:

• *непротиворечивой* (не должно быть данных, противоречащих друг другу);

• *неизбыточной* (следует избегать ненужного дублирования информации в базе, избыточность может привести к противоречивости – например, если какие – то данные изменяют, а их копию в другой части базы забыли изменить);

• *целостной* (все данные должны быть связаны, не должно быть ссылок на несуществующие в базе данные)

Реляционная модель баз данных была предложена Эдгаром Коддом в конце 70-х годов. В рамках этой модели база данных представляет собой набор таблиц, связанных друг с другом отношениями. При достаточной простоте (а значит, и удобстве реализации на компьютере) данная модель обладает гибкостью, позволяющей описывать сложно структурированные данные. Кроме того, для этой модели достаточно глубоко проработано теоретическое обоснование, что также даёт возможность эффективнее использовать компьютер при создании базы

данных и работе с ней. В плане правил связи в реляционной модели реализуется отношение «один–ко–многим» связи между таблицами. Это значит, что одной записи в главной таблице соответствует несколько записей в подчинённой таблице (в том числе может не соответствовать ни одной записи). Другие типы связей: «один-к-одному», «много-к-одному» и «много-ко-многим» - можно свести к данному типу «один-ко-многим». Реляционные базы данных состоят из связанных таблиц.

Таблица представляет собой двумерный массив, в котором хранятся данные. Столбцы таблицы (в рамках принятых обозначений БД) называются полями, строки – записями. Количество полей таблицы фиксировано, количество записей – нет. Фактически таблица – нефиксированный массив записей с одинаковой структурой полей в каждой записи. Добавить в таблицу новую запись не составляет труда, а то время как добавление нового поля влечёт за собой рестрктуризацию всей таблицы и может вызвать определённые трудности. В качестве значений полей в записях могут храниться числа, строки, картинки и т.д. Таблицы баз данных хранятся на жёстком диске (на локальном компьютере или на сервере баз данных – в зависимости от типа БД). Одной таблице соответствуют обычно несколько файлов – один основной и несколько вспомогательных. Тонкости организации таблиц зависят от используемого формата (dBase, Paradox, InterBase, Microsoft Access и т.д.)

Ключ – поле или комбинация полей таблицы, значения в которых однозначно определяют запись. Ключ потому так и называется, что, имея значения ключевых полей, можно однозначно получить доступ к нужной записи. Таким образом, ключи чрезвычайно полезны для связи таблиц. Записывая значения ключа в отведённые поля подчинённой таблицы и тем самым, задавая ссылку, обеспечиваем связь двух записей – записи в главной таблице и записи в подчинённой таблице. В одной записи подчинённой таблицы может находиться и несколько ссылок на записи главной таблицы. Например, в школьном журнале может быть таблица – список дежурств, где в каждой записи содержатся фамилии и имена (ключ их двух полей) нескольких дежурных. Так осуществляется связь различных записей главной таблицы и реализуется достаточно сложная структура данных. В школьной практике в качестве ключевых полей используются имена и фамилии, но в БД лучше отводить специальные ключевые поля – индивидуальные номера (коды) записей. Это гарантированно уберегает от возможных проблем с однофамильцами. В школе же, где не требуется такая компьютерная чёткость, появление в одном классе двух учеников с одинаковыми именами и фамилиями – очень редкое событие, поэтому можно простить подобное техническое упущение. Кроме

связывания, ключи могут использоваться для прямого доступа к записям, ускорения работы с таблицей.

Индекс – поле, так же, как и ключ, специально выделенное в таблице, данные в котором, однако, могут повторяться. Они также служат для ускорения доступа и, кроме того, для сортировки и выборок.

Нормальные формы были придуманы, скорее, для автоматизации процесса создания баз данных, нежели как руководство тем, кто создаёт их вручную (автоматическое проектирование больших баз данных может производиться с помощью специальных систем программ – средств (CASE). Реально при ручной разработке проектировщик сразу же задумывает необходимую структуру, планирует нужные таблицы, а не идёт от одной большой таблицы. Нормальные формы фактически формализуют интуитивно понятые требования к организации данных, помогая, прежде всего, избежать избыточного дублирования данных.

Первая нормальная форма:

• *информация в полях неделимая* (к примеру, имя и фамилия должны быть разными полями, а не одним);

• *в таблице нет повторяющихся групп полей*

Вторая нормальная форма:

• *выполнена первая форма;*

• *любое неключевое поле однозначно идентифицируется ключевыми полями* (фактически, требование наличия ключа)

Третья нормальная форма:

• *выполнена вторая форма*

• *неключевые поля должны однозначно идентифицироваться только ключевыми полями* (это значит, что данные, не зависящие от ключа, должны быть вынесены в отдельную таблицу)

Требование третьей нормальной формы имеет тот смысл, что таблицу с полями (Имя, Фамилия, Класс, Классный руководитель) необходимо разбить на две таблицы (Имя, Фамилия, Класс) и (Класс, Классный руководитель), поскольку поле Класс однозначно определяет поле Классный руководитель (а согласно третьей

форме, однозначно определять должны только ключи).

Для более глубокого понимания тонкостей проведения операций с записями в таблицах необходимо иметь понятия о способах доступа, транзакциях и бизнес-правилах.

Способы доступа определяют, как технически производятся операции с записями. Способы доступа выбираются программистом во время разработки приложения. Навигационный способ основан на последовательной обработке нужных записей поодиночке. Он обычно используется для небольших локальных таблиц. Реляционный способ основан на обработке сразу набора записей с помощью SQL-запросов. Он используется для больших удалённых БД.

Транзакции определяют надёжность выполнения операций по отношению к сбоям. В транзакцию объединяется последовательность операций, которая либо должна быть выполнена полностью, либо не выполнена совсем. Если во время выполнения транзакции произошёл сбой, то все результаты всех операций, входящих в неё отменяются. Это гарантирует то, что не нарушается корректность базы данных даже в случае технических (а не программных) сбоев.

Бизнес-правила определяют правила проведения операций и представляют механизмы управления БД. Задавая возможные ограничения на значения полей, они также вносят свой вклад в поддержание корректности базы. Несмотря на возможные ассоциации с бизнесом как коммерцией, бизнес-правила не имеют к нему прямого отношения и просто являются правилами управления базами данных.

Корректная БД:

- *неизбыточная;*

- *непротиворечивая;*

- *целостная*

Реляционная БД:

- *таблицы;*

- *связи между таблицами с помощью ключей*

Таблица:

- поля (столбцы) – фиксированы;

- записи (строки) – легко добавляются и удаляются

Ключ:

- однозначно определяет запись

Ключи и индексы:

- служат для связи таблиц, прямого доступа, ускорения обработки и т.п.

Нормальные формы:

- служат для борьбы с избыточностью данных;

- много требуют, но из самых благих побуждений

Способы доступа:

-навигационный;

- реляционный

Защита корректности БД:

- транзакции – техническая защита

- бизнес-правила – логическая защита

Л И Т Е Р А Т У Р А :

1. Информатика. Учебное пособие для среднего профессионального образования (+CD)/Под общ. ред. И.А. Черноскутовой – СПб.: Питер, 2005. – 272 с.: ил. стр. 24 - 25

2. Информатика. Учебное пособие для студ. пед. вузов /А.В.Могилёв; Н.И.Пак, Е.К.Хённер; Под ред. Е.К.Хённера. – М., 1999. - 816 с стр. 185 - 187

3. Информатика. Учебник. – 3-е перераб. изд./Под ред. проф. Н.В.Макаровой. – М.: Финансы и статистика, 2000. – 768 с.: ил.