Базы данных (БД) и СУБД ACCESS

Литература

1. **Информатика: Учеб. пособие для студентов пед. вузов / А. В. Могилёв, Н.И. Пак, Е.К. Хеннер; Под ред. Е.К. Хеннера – М.: Изд. центр “Академия”, 2000. – 816 с.**
2. **Информатика: Учебник.—3-е переб.изд./Под ред. Н. В. Макаровой. – М.: Финансы и статистика, 2002. – 768 с.**
3. **Информатика. Базовый курс. 2-е издание /Под ред. С.В. Симоновича. - СПб.: Питер, 2005. – 640 с.**
4. **Петров В.Н. Информационные системы. – СПб.: Питер, 2002. — 688 с.**
5. **Хабрейкен, Джо. Microsoft Office 2003. Всё в одном.: Пер. с англ. — М.: Вильямс, 2006. — 864 с.**
6. **Далаа С.М. Основные принципы работы в Microsoft Access. – Кызыл: Изд-во ТывГУ, 2008. – 63 с.**

ВВЕДЕНИЕ

Программное обеспечение за время своего существования претерпело большие изменения: от программ, способных выполнять простейшие логические и арифметические операции до сложных систем управления организацией.

Сегодня управление организацией без компьютера просто немыслимо. Компьютеры давно и прочно вошли в такие области управления, как бухгалтерский учет, отдел кадров и т.п. Как правило, эти области связаны с большими массивами данных, которые периодически меняются.

Создание программ в таких областях требует решения задачи разработки базы данных, предназначенной для хранения и изменения информации.

ОСНОВНЫЕ ПОНЯТИЯ

Существует много определений базы данных.

В широком смысле слова *база данных* – это совокупность сведений о конкретных объектах реального мира в какой-либо предметной области. Под предметной областью принято понимать часть реального мира, подлежащего изучению для организации управления и в конечном счете автоматизации, например, вуза, фирмы и т. д.

База данных – это совокупность записей различного типа, содержащая перекрестные ссылки. Это определение повторяет одно из множества классических определений баз данных, которое предложено международной Ассоциацией по языкам систем обработки данных *КОДАСИЛ*.

ОСНОВНЫЕ ПОНЯТИЯ

Создавая базу данных, пользователь стремится упорядочить информацию по различным признакам и быстро извлекать данные по различным критериям.

Сделать это возможно, только если данные структурированы.

Структурирование – это введение соглашений о способах представления данных. Неструктурированными называют данные, записанные, например, в текстовом файле:

N1 Сат Иван Петрович Дата рождения 1 января 1990 г.
N2 Петрова Аяна Маадыровна Дата рождения 10 мая 1991 г. …

В таком виде сложно организовать поиск необходимых данных, а упорядочить практически невозможно.

ОСНОВНЫЕ ПОНЯТИЯ

После структуризации этой информации она будет выглядеть следующим образом:

Номер	Фамилия	Имя	Отчество	Дата рождения
1	Сат	Иван	Петрович	01.01.1990
2	Петрова	Аяна	Маадыровна	10.05.1991
…	…	…	…	…

В современной технологии баз данных предполагается, что создание базы данных, ее поддержка и обеспечение доступа пользователей к ней осуществляются централизованно с помощью специального программного инструментария – *системы управления базами данных* (*СУБД*).

СУБД делятся на *одно-*, *двух-* и *трёхзвенные*.

ОСНОВНЫЕ ПОНЯТИЯ

В *однозвенной СУБД* используется единственное звено (*клиент*), обеспечивающее необходимую логику управления данными и их визуализацию.

В *двухзвенной СУБД* (*клиент* и *сервер базы данных*) значительную часть логики управления берет на себя *сервер базы данных*, в то время как клиент в основном занят отображением данных в удобном для пользователя виде.

В *трехзвенной СУБД* (*клиент, сервер базы данных и сервер приложений*) используется промежуточное звено – *сервер приложений*, являющееся посредником между *клиентом* и *сервером* базы данных.

В зависимости от места расположения отдельных частей *СУБД* различают *локальные* и *сетевые*.

Все части *локальной СУБД* находятся на компьютере пользователя базы данных.

ОСНОВНЫЕ ПОНЯТИЯ

Непременным атрибутом сетевых *СУБД* является сеть, обеспечивающая аппаратную связь компьютеров и делающая возможной корпоративную работу множества пользователей с одними и теми же данными.

Ядром любой базы данных является модель данных.

Модель данных – это совокупность структур данных и операций их обработки. С помощью модели данных могут быть представлены объекты предметной области и взаимосвязи между ними.

На сегодня существует несколько моделей баз данных. Мы будем рассматривать *реляционную модель* баз данных.

В *реляционной модели* база данных состоит из наборов (таблиц) записей одинаковой структуры, в которых хранятся данные.

ОСНОВНЫЕ ПОНЯТИЯ

Понятие реляционной базы данных тесно связано с такими понятиями структурных элементов, как *поле*, *запись*, *файл* (таблица).

Поле – это элементарная единица логической организации данных.

Для описания поля используются следующие характеристики:

- *имя* (например, *Фамилия*, *Имя* и т.д.);
- *тип* (например, *символьный*, *числовой* и т.д.);
- *длина* (например, 15 байт, причем это максимально возможное число для данного поля);
- *точность для числовых полей* (например, два десятичных знака для отображения дробной части числа).

Запись – совокупность логически связанных полей.

Экземпляр записи – отдельная реализация записи, содержащая конкретные значения ее полей.

ОСНОВНЫЕ ПОНЯТИЯ

Файл (*таблица*) – совокупность экземпляров записей одной структуры.

В структуре записи файла указывается поле, значения которого являются *ключами*. Его еще называют *первичным ключом*.

Это поле должно содержать уникальный идентификатор, позволяющий отличить одну запись от другой.

В качестве ключей обычно используется цифровое значение. Так как текстовый ключ занимает много места (для ключа-числа достаточно нескольких байтов), что понижает эффективность работы СУБД.

В большинстве реляционных СУБД ключи реализуются с помощью объектов, называемых *индексами*. *Индекс* представляет собой указатель на данные, размещенные в реляционной базе данных.

ОСНОВНЫЕ ПОНЯТИЯ

В реляционных базах данных по определению между отдельными таблицами должны существовать связи. При установлении связи между двумя таблицами одна из них будет *главной (master)*, а вторая – *подчиненной (detail)*.

Отличие этих двух таблиц друг от друга в том, что при работе с главной таблицей можно обращаться к подчиненной таблице, но при этом будут доступны только те записи, которые связаны с текущей записью главной таблицы.

Различают четыре типа связей между таблицами реляционной базы данных:

- *один к одному* – каждой записи одной таблицы соответствует только одна запись другой таблицы;

ОСНОВНЫЕ ПОНЯТИЯ

- *один ко многим* – одной записи главной таблицы могут соответствовать несколько записей подчиненной таблицы;
- *многие к одному* – нескольким записям главной таблицы может соответствовать одна та же запись подчиненной таблицы;
- *многие ко многим* – одна запись главной таблицы связана с несколькими записями подчиненной таблицы, а одна запись подчиненной таблицы связана с несколькими записями главной таблицы.

Использование реляционных баз данных возможно только при наличии эффективного языка управления реляционными данными. Наибольшее распространение получил язык *SQL* — *Structured Query Language* (*структурированный язык запросов*). Команды языка *SQL* обычно подразделяются на несколько групп:

- *язык определения данных* (*ЯОД*), команды которого используются для создания и изменения структуры объектов базы данных, и язык манипулирования;
- *язык манипулирования данными* (*ЯМД*), команды которого используются для манипулирования информацией, содержащейся в объектах базы данных;
- *язык управления данными* (*ЯУД*), команды которого предназначены для управления доступом к информации, хранящейся в базе данных;

- команды, предназначенные для формирования запросов к базе данных (*запрос* — это обращение к базе данных для получения соответствующей информации);
- команды управления транзакциями (*транзакция* — это последовательность операций над базой данных, рассматриваемых СУБД как единое целое).

Проектирование БД

Теперь переходим к проектированию таблиц базы данных. Нужно выяснить:

1. Какова структура таблицы, т.е. с*колько полей будет в таблице и какого типа?*
2. *Какое поле будет ключевым?*
3. *Какие поля будут связывать главную таблицу с подчиненной?*

Это нужно для того, чтобы исключить избыточность информации и противоречивость хранимых данных в базе, что упрощает управление ими.

Рассмотрим это на конкретном примере. Поставим задачу – создать базу данных, с помощью которой можно было бы вести учет книг для студентов и контроль за читателями этих книг.

Проектирование БД

Можно, не долго думая, создать в базе данных одну таблицу по данной задаче. Но этот подход приведет к *избыточности данных* в базе. Например, в нашем случае информация по студентам будет повторяться, так как один студент может читать несколько книг. А это в случае настоящей базы данных (порядка нескольких миллионов записей) приводит к неэффективной работе пользователя с базой данных.

Мы пропустили очень важный процесс в проектировании базы данных. Это процесс *нормализации данных*.

Нормализация - это процесс реорганизации данных для ликвидации повторяющихся групп и иных противоречий, чтобы привести данные к виду, позволяющему непротиворечиво и корректно работать с ними.

Окончательная цель нормализации сводится к получению такого проекта базы данных, в котором каждый факт появляется лишь в одном месте, то есть исключена избыточность информации.

Избыточность информации устраняется не столько с целью экономии памяти ЭВМ, сколько для исключения возможной противоречивости хранимых данных и упрощения управления ими.

Использование ненормализованных таблиц может привести к следующим проблемам:

- *избыточности данных* (в нескольких записях таблицы базы данных повторяется одна и та же информация);
- *аномалия обновления*, которая тесно связана с избыточностью данных (например, если при обновлении данных в таблице исправлены не все записи, то возникнет несоответствие информации);

• *аномалия удаления* (например, в нашем случае при удалении читателя из таблицы мы можем также удалить и книгу, которую он взял);
• *аномалия ввода*. Эта ситуация возникает при добавлении в таблицу новых записей, когда для некоторых полей таблицы заданы ограничения *NOT NULL* (не пусто). В нашем случае, например, при появлении новой книги в таблице поле «Кто взял книгу» обязательно должно быть заполнено, хотя книга еще не взята.

Очевидно, что эти аномалии крайне нежелательны. Чтобы свести к минимуму возможность появления такого рода аномалий, и используется нормализация.

Теория нормализации основана на концепции *нормальных форм*. Каждой нормальной форме соответствует некоторый определенный набор ограничений. В теории реляционных баз данных обычно выделяется следующая последовательность нормальных форм:

1) первая нормальная форма;
2) вторая нормальная форма;
3) третья нормальная форма;
4) нормальная форма Бойса-Кодда;
5) четвертая нормальная форма;
6) пятая нормальная форма или нормальная форма проекции-соединения;

Основные свойства нормальных форм:

- каждая следующая нормальная форма в некотором смысле лучше предыдущей;
- при переходе к следующей нормальной форме свойства предыдущих нормальных форм сохраняются.

Теперь вернемся к нашей задаче. Нужно создать две таблицы: таблица книг и таблица студентов. Пусть первая таблица будет главной, а вторая – подчиненной.

Тогда связь между таблицами будет «один ко многим», так как одна книга может иметь несколько читателей.

Перейдем к проектированию таблиц базы данных. Нужно выяснить:

1. *Какова структура таблицы, т.е. сколько полей будет в таблице и какого типа?*
2. *Какое поле будет ключевым?*
3. *Какие поля будут связывать главную таблицу с подчиненной?*

Начнем с таблицы книг. В ней должны быть следующие сведения: номер книги (целый), название книги (строковый), автор книги (строковый), издательство (строковый), год издания (целый), стоимость книги (денежный). *Номер книги* будет ключевым полем в таблице книг.

В таблице студентов-читателей книг должны быть сведения: номер читателя (целый), фамилия с инициалами (строковый), номер книги (целый), даты получения и возврата книги (типа даты), потерял ? (логический), примечание (строковый). *Номер читателя* будет ключевым полем в таблице читателей.

Поля «Номер книги» в таблице книг и «Номер книги» в таблице читателей будут связывать эти таблицы.

Таблица *Knigi* связана с таблицей *Man* отношением *один ко многим*.

Из всего этого вытекает следующий вывод, чтобы создать эффективную базу данных, нужно предварительно спроектировать ее структуру. А для проектирования базы все данные должны быть структурированы в зависимости от поставленной задачи. Для этого необходимо использовать все возможности, которые есть у вас.

Примечания

1. Не стоит давать русскоязычные имена таблицам и полям. Это создаст проблемы при использовании *SQL*-запросов.
2. В именах полей полезно ставить префикс из одной или двух букв имени таблиц, то есть имена полей в таблице *Knigi* начать с буквы “K”, а в *Man* – с “M” и т.д. Это исключит вероятность того, что вы случайно назовете поле с одним из зарезервированных служебных слов, а это может привести к ошибке. При этом легче будет узнать, какой таблице принадлежит данное поле.
3. Желательно бы иметь структуру базы данных в печатном (на бумаге) или электронном виде (файле). Чтобы знать, где и какие данные хранятся и как они взаимосвязаны между собой.

СУБД ACCESS

Одним из эффективнейших средств работы с базами данных является СУБД — система управления базами данных. Для персональных компьютеров широко распространены СУБД типа DBASE (DBASE III, IV, FoxPro, Paradox), Clipper, Clarion. Эти СУБД ориентированы на однопользовательский режим работы с базой данных и имеют очень ограниченные возможности. В связи с развитием компьютерных сетей, в которых персональные компьютеры выступают в качестве клиентов, новые версии СУБД все в большой степени включают в себя возможности работы с базой данных, которая расположена на сервере компьютерной сети.

В последнее время среди СУБД стали популярными Access, Lotus, Oracle и MySQL.

ACCESS может работать с файлами трех видов:

1. *Файл базы данных* Microsoft Access;
2. *Файл проекта Microsoft Access* — это файл, который не содержит данные, но имеет подключение к базе данных Microsoft SQL Server;
3. *Файл страницы доступа к данным* — это файл, с помощью которого можно организовать ввод данных через Интернет или интранет для создания отчетов по этим данным или анализа их.

Мы будем работать только с файлами баз данных.

База данных в ACCESS представляет собой *реляционную базу данных*, т.е. это совокупность двумерных таблиц. Количество строк и столбцов каждой таблицы зависит от решаемой задачи и возможностей компьютера.

Строки таблицы называются *ЗАПИСЯМИ* и нумеруются натуральными числами.

Столбцы таблицы называются *ПОЛЯМИ*. Каждое поле имеет свое имя, состоящее из букв, цифр и знака подчеркивания. Запрещено использовать символы:

! . [ ] ‘

Имя поля должно быть *уникальным*, то есть не совпадать с другими именами и служебными словами, использующиеся в среде ACCESS. Длина имени поля не должна превышать 64 символов.

Имена полей и номер записи используются для определения местонахождения данных.

В поле таблицы хранится однотипная информация. В ACCESS существуют следующие типы полей:

- *Текстовый*. Это текст или комбинация текста и чисел, например, номера телефонов, номенклатурные номера или почтовый индекс. Длина текстового поля - до 255 символов.

- *Числовой*. Это числовые данные, используемые для математических вычислений, за исключением вычислений, включающих денежные операции (используйте денежный тип).
- *Дата/время*. Это даты и время, относящиеся к годам с 100 по 9999, включительно.
- *Денежный*. Это значения валют. Денежный тип используется для предотвращения округлений во время вычислений. Предполагает до 15 символов в целой части числа и 4 - в дробной.
- *Счетчик*. Автоматическая вставка последовательных (отличающихся на 1) или случайных чисел при добавлении записи. Редактирование этого поля невозможно.
- *Логический*. Поля такого типа содержат только одно из двух возможных значений, таких как “Да/Нет”, “Истина/Ложь”, “Включено/Выключено”.

- *Поле объекта OLE*. Это объекты (например, документы Microsoft Word, электронные таблицы Microsoft Excel, рисунки, звуки и другие данные), созданные в других программах, использующих протокол OLE. Объекты могут быть связанными или внедренными в таблицу Microsoft Access. Размер поля - до 1 гигабайта (ограничено объемом диска).
- *Гиперссылка*. Строка, состоящая из букв и цифр, и представляющая адрес гиперссылки.
- *Мастер подстановок*. Создает поле, в котором предлагается выбор значений из списка, или из поля со списком, содержащего набор постоянных значений или значений из другой таблицы. Выбор этого параметра в списке в ячейке запускает мастера подстановок, который определяет тип поля.

Файл базы данных ACCESS, кроме таблиц базы данных, содержит также следующие объекты:

- *запрос* — это совокупность данных из нескольких таблиц базы, удовлетворяющих некоторому условию;
- *форма* — это совокупность данных из нескольких таблиц базы, сформированная для редактирования информации;
- *отчет* — это совокупность данных из нескольких таблиц базы, сформированная для вывода на печать;
- *страница* — это совокупность данных из нескольких таблиц базы, доступная для всех пользователей в локальной или глобальной сетях;
- *макрос* — это набор из одной или более макрокоманд, позволяющих выполнять определенные операции, такие как открытие форм или печать отчетов. Макрокоманда — это стандартная подпрограмма, которая написана на языке VBA (Visual Basic for Applications);
- *модуль* — это программа на языке VBA.

СОЗДАНИЕ БАЗЫ ДАННЫХ В ACCESS

Создание базы данных в ACCESS делится на 2 этапа:

1. Создание файла базы данных.
2. Создание таблицы.

Создание файла базы данных:

- Щелкнуть по опции *Файл* меню ACCESS.
- Щелкнуть по опции **Создать…** (в рабочем окне ACCESS появится окно *Создание файла*).

В окне *Создание файла* щелкнуть по опции *Новая база данных*.

В диалоговом окне *Файл новой базы данных* определить папку, тип файла и дать имя файла. Стандартное расширение файла базы данных ACCESS - это *mdb* . Затем щелкнуть по кнопке *Создать*.

Если все правильно сделали, то появится окно созданной базы данных (например, *baza : база данных (формат Access 2002-2003*), где *baza* - имя базы данных), в котором можно работать с таблицами, запросами, формами, отчетами, макросами и модулями Visual Basic.

Данное окно состоит из трех частей.

Первая часть — это верхняя строка окна, в которой расположены кнопки.

Вторая часть — это левый столбец окна, в которой расположены три вкладки: *Объекты, Группы, Избранное*.

Третья часть - это основная часть окна, в которой располагаются объекты базы данных.

Создание таблицы

Чтобы создать таблицу, нужно сначала определить структуру таблицу: сколько полей должно быть, какого типа и формата они должны быть.

Сразу договоримся что все объекты будем создавать в режиме *Конструктора*, т.е. самостоятельно создавая и редактируя объекты.

Рассмотрим это на примере таблицы *Книги*:

1. В диалоговом окне *: база данных (формат Access 2002-2003*) щелкнуть по объекту *Таблица*.
2. Щелкнуть по опции *Создание таблицы в режиме конструктора*.
3. В диалоговом окне *Таблица1: таблица* определяем:
   - имя первого поля. С клавиатуры вводим имя поля в первой колонке таблицы.

- тип этого поля. Для этого нужно щелкнуть по колонке с именем *Тип данных*, появится кнопка ▼, щелчок по которой вызывает список типов полей. Внизу диалогового окна появятся общие свойства поля выбранного типа (такие как *Размер поля*, *Формат поля* и другие), которые можно изменять.

- описание поля. Это можно не делать.

- имя второго поля во второй строке данной таблицы.

- . . .

После определения всех полей таблицы нужно выйти из режима конструктора (щелчок по кнопке X) или перейти в режим таблицы (первая кнопка *Вид* на панели кнопок). Режим таблицы предназначен для ввода и редактирования данных таблицы. Перед тем, как выйти из режима конструктора, ACCESS предложит подтвердить создание таблицы заданием имени таблица (по умолчанию таблица получает имя *Таблица1* или *Таблица2* и т.д.).

Примечания.

1. В окне *baza : база данных (формат Access 20002-2003)* вы можете не только создать таблицу, но и переименовать или уничтожить ее, просмотреть ее свойства с помощью контекстно-зависимого меню.
2. При создании таблицы рекомендуем имена полей составлять из латинских букв и цифр, так как имена полей используются в выражениях для вычисляемых полей в запросах, формах, отчетах и страницах доступа, а также в макросах и модулях базы данных. Это уменьшит вероятность появления ошибок. Обычно при просмотре данных базы в режиме таблицы в заголовке таблицы выводятся имена полей, которые обычно записаны латинскими буквами. Для замены имени поля в заголовке таблицы при просмотре таблицы нужно использовать свойство поля *Подпись*.

ПОДСТАНОВКА ДАННЫХ В ТАБЛИЦЕ

Для удобства ввода данных в поле таблицы можно заранее заготовить список часто вводимых данных. Например, в таблице студентов имеется поле, содержащее года рождения студентов. Можно предположить, что большинство студентов имеют года рождения, отличающиеся друг от друга самое большее на 5 лет, то будут встречаться года рождения: 1990, 1991, 1992, 1993 и 1994. Эти числа можно занести в структуру таблицы студентов следующим образом:

1. Открыть данную таблицу с помощью конструктора.
2. Выбрать указанное поле. В свойствах этого поля выбрать вкладку *Подстановка*.
3. Щелкнуть мышкой в поле опции *Тип элемента управления* и вызвать список значений этой опции щелчком мышки по кнопке ▼ и установить значение «Поле со списком».

4. В опции *Тип источника строк* выбрать значение «Список значений».

5. В опции *Источник строк* с клавиатуры ввести данные года рождения через точку с запятой:

1990; 1991; 1992; 1993; 1994

6. Сохранить данную структуру таблицу и открыть таблицу для работы с данными.

Для проверки щелкните по указанному полю. Должна появиться кнопка ▼, которая должна вызывать наш список годов рождений. После выбора нужного года рождения этот год должен появиться в этом поле.

Можно эти года рождения хранить в отдельной таблице *God*. Например, пусть это будет таблица (она должна состоять из одного поля, в котором хранятся данные года рождения). Тогда в опции *Тип источника строк* выбрать нужно выбрать значение «Таблица или запрос», а в опции *Источник строк* выбрать таблицу *God* с помощью кнопки ▼.

Организация фильтра в таблице

Для удобства редактирования записей (если их много) следует установить фильтр на записи, чтобы отобрать те записи, которые нужны для редактирования. Это можно сделать через панель кнопок:

- щелкнуть по кнопке *Изменить фильтр* на панели кнопок ACCESS (в таблице все записи исчезнут и будет одна пустая запись);
- под именем поля, по которому вы будете создавать фильтр, введите выражение для отбора записей;
- то же самое можно с другим полем, если это нужно;
- после создания фильтра нужно применить его щелчком по кнопке *Применение фильтра*.

После работы с отфильтрованными записями следует отменить фильтр с помощью опций *Записи*, *Удалить фильтр* меню ACCESS.

Выражения для фильтра в таблице

выражение	Значение поля
<>0	отлично от нуля.
0 Or >100	либо равно 0, либо больше 100.
Like "K??"	содержит три символа и начинается с буквы K.
Like "A*"	должно начинаться с буквы A, остальные буквы любые
<#1/1/96#	содержит даты, предшествующие 1996.
>= #1/1/97# And < #1/1/98#	содержит даты в пределах 1997 года.

Связанные таблицы

Пусть мы создали две таблицы: первая таблица содержит данные по читателям, вторая таблица — данные книг, которые прочитали или читают эти читатели. Чтобы связать эти таблицы в ACCESS нужно:

1. Определить, какая таблица будет главной, а какая — подчиненной. Пусть таблица читателей—главная, а таблица книг— подчиненной.
2. В подчиненной таблице обязательно должно быть поле, содержимое которого должно совпадать с содержимым полем-ключом главной таблицы. Типы этих полей должны быть одинаковыми! Если поле-ключ типа *Счетчик*, то соответствующее поле в подчиненной таблице должно быть целого типа.
3. Создать связь между этими таблицами с помощью опции меню ACCESS: *Сервис*, *Схема данных.* Следующим образом:

- В появившемся диалоговом окне *Добавление таблицы* выбрать таблицы, которые нужно связать, с помощью кнопки *Добавить*. После этого диалоговое окно закрыть.
- В окне *Схема данных* выбрать в главной таблице поле-ключ и протяжкой мыши перенести это поле на подчиненную таблицу, а именно на поле, которое создано специально для этого.
- В окне *Изменение связей* ставите галочку щелчком мыши в опции *Обеспечение целостности данных*.
- После этого щелкаете мышкой по кнопке *Создать*.

Таким образом, мы создали связь между таблицами типа «один ко многим». В окне *Схема данных* между выбранным полями данных таблиц будет нарисована эта связь. Эту связь можно удалить, вызвав контекстное меню щелчком по правой кнопке мышки. И снова создать новую связь, выбрав другие поля таблиц. При закрытии окна появится диалоговое окно о сохранении макета «Схема данных».

Практические советы

1. Перед заполнением связанных таблиц данными, нужно обязательно создать связь. Затем заполнение начинать надо с главной таблицы, так как можно одновременно тогда заполнять и подчиненную таблицу. Обратите внимание на символ «+», который будет стоять перед номером записи. Если щелкнуть мышкой по нему, то появится подчиненная таблица.
2. Чтобы удалить запись в главной таблице, нужно сначала удалить все записи в подчиненной таблице, связанные с этой записью. А затем удалять эту запись. Если при создании связи между таблицами вы отметили галочкой опцию *каскадное удаление связанных записей* в окне *Изменение связи*, то вы можете сразу уничтожить запись в главной таблице и автоматически тогда уничтожатся связанные с ней записи в подчиненной таблице.

Работа с запросом

ЗАПРОС – представление данных из нескольких таблиц в удобном виде. Запросы обеспечивают быстрый и эффективный доступ, хранящимся в таблице.

ДЛЯ СОЗДАНИЯ ВЫЧИСЛЯЕМОГО ПОЛЯ В ЗАПРОСЕ следует в окне *Запрос1 : запрос на выборку* (режим конструктора), а именно в опции *Поле*: ввести следующую конструкцию:

<имя выч.поля> **:** <выражение>

Например:

Пенсионный фонд : [zarpl] * 0,1

где [zarpl] – имя поля в таблице.

Т.к. в запросе в заголовке таблицы выводятся имена полей. Можно сменить их на русские имена, используя конструкцию вычисляемого поля. Например:

Фамилия, имя, отчество : [FIO]

где [FIO] – имя поля в таблице.

Организация фильтра в наборе данных

Примеры выражений фильтров:

1. *Avtor='Макарова'* - отбираются все записи, у которых поле *Avtor* имеет значение 'Макарова'
2. *Price*Quan>100* - будут выбраны записи, для которых произведение значения поля *Price* на значение поля *Quan* больше 100
3. *Data<='31.12.2001' and Name='A*'* - выбираются записи, у которых значение поля *Data* не превышает '31.12.2001' и при этом значение поля *Name* начинается с буквы «А».

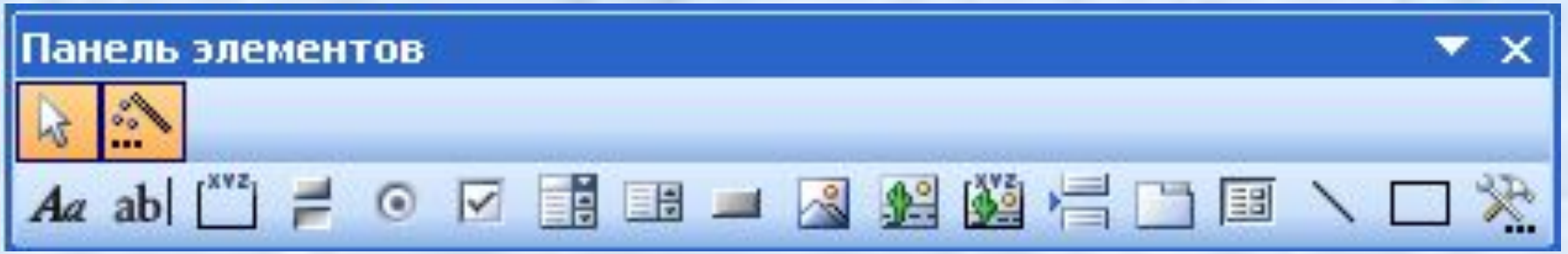
С помощью свойства *FilterOptions* можно задать параметры фильтра. Это свойство имеет значения:

foCaseInsensitive – регистр букв не будет учитываться при фильтрации;

foNoPartialCompare – обеспечивает проверку на полное соответствие значения выражения и содержимого поля.

Работа с формой

ФОРМА – обработка данных таблиц и запросов в удобном виде. Преимущество формы для ввода и редактирования данных состоит в простоте и наглядности. В режиме формы вы сможете в полной мере воспользоваться возможностями WINDOWS (различные масштабируемые шрифты, графика и т.д.). Кроме того, вы можете создавать необходимые для решения своей задачи элементы формы:



Aa “надпись”	Для создания надписи в форме.
ab| “поле”	Для создания поля с надписью в форме С помощью этого элемента вы можете создать ВЫЧИСЛЯЕМОЕ ПОЛЕ В ФОРМЕ. Для этого внутри поля следует ввести выражение в следующем формате: = <выражение> Например: = year[*data*]+5 где *data* – имя поля в источнике данных, содержащее данные типа *дата/время*, year — функция, возвращающая год из аргумента типа *дата/время*. Имена полей в выражениях всегда заключаются в квадратные скобки.
 “выключатель”	Данный элемент используется в форме для работы с логическим полем, который должен принимать значение из двух возможных.
 “переключатель”	Элемент “переключатель” также предназначен для работы с логическим полем. Появление в нем жирной точки соответствует логическому значению “Истина” , а отсутствие его - значению “Ложь”.
 “флажок”	Элемент “флажок” тоже предназначен для работы с логическим полем.