|  |
| --- |
|  |

|  |
| --- |
| **Лабораторная работа  «Разработка форм»** |
| *Лабораторный практикум по базам данных* |
|  |
| 1. Простые формы 2. Иерархические формы 3. Синхронизированные формы 4. Формы, состоящие из нескольких страниц или вкладок 5. Всплывающая форма и пользовательское диалоговое окно |
|  |
| **ЧувГУ, каф. Вычислительной техники** |
| **17.06.2022** |
|  |

# Лабораторная работа «Разработка форм»

После окончательного создания приложения пользователь, как правило, не работает непосредственно с таблицами базы данных. Разработчик приложения часто ограничивает полностью или частично непосредственный доступ пользователя к таблицам. Интерфейс пользователь-база данных обеспечивается формами. Эффективность работы с базой данных во многом зависит от качества разработки форм.

Все формы разделим на две группы:

* формы для ввода данных
* формы для анализа и обработки данных

При разработке форм первой группы необходимо учитывать структуру первичных документов, так как именно они являются источником исходных данных. Это означает, что

* строить формы ввода, исходя только из структуры базы данных, было бы заблуждением. Другими словами, для каждой таблицы – своя форма ввода, самое простое, но не всегда наилучшее решение. Например, если бланк заказа включает в себя список заказываемых товаров, то форма ввода кроме основных данных по заказу будет содержать подчиненную форму списка товаров.
* при размещении полей ввода в форме необходимо учитывать порядок расположения аналогичных полей в исходном первичном документе. Это значительно облегчит пользователю ввод данных. Например, паспортные данные в форме должны по возможности следовать в том же порядке, что и в паспорте.

Формы второй группы предназначены только для анализа и обработки данных, поэтому в них рекомендуется

* преобразовать поля со списком в простые поля,
* запретить все операции, связанные с преобразованием, изменением данных и вводом новых записей.

Для распознавания класса формы рекомендуется в название формы ввести префиксы «фвв» и «фад» для форм ввода и анализа данных соответственно. Например, фвв\_Служащий, фад\_Рейтинг.

Можно рекомендовать в практических приложениях предусмотреть сначала ввод в базу справочных данных, а затем данных плановых и оперативно-учетных документов.

# Пример из БД «Журнал преподавателя».

*Справочная информация*: таблицы «ПРЕДМЕТ», «СТУДЕНТ», «КАФЕДРА», …

*Плановая информация*: таблицы «НАГРУЗКА», «ЗАНЯТИЯ»

*Оперативно-учетная информация: таблицы «ПОСЕЩАЕМОСТЬ», «УСПЕВАЕМОСТЬ»*

**Примечание.** Приведенные вышепредложения основаны на личном опыте и потому носятрекомендательный характер. Однако отклонения от данных рекомендаций должны быть обоснованы.

С предложениями по установке свойств формы можно ознакомиться в приложении «[Свойства форм для анализа данных](#п2)».

С классификацией создаваемых типов форм можно ознакомиться в прилагаемой ниже статье «[Типы форм](#_Типы_форм)».

## Что надо знать

## ОСНОВЫ РАБОТЫ С ФОРМАМИ

1. Создание форм (**🕮** Создание БД и объектов и работа с ними \Объекты БД \Работа с формами)
   1. Режимы формы (**🕮** Режимы )

 Помните, что для любой формы можно изменить ее представление, определяющее режим отображения данных. В окне свойств формы можно выбрать один из пяти вариантов: *Одиночная**форма*(Single Form), *Ленточные формы*(Continuous Form), *Режим таблицы*(Datasheet), *Сводная**таблица*(PivotTable) и *Сводная диаграмма*(PivotChart).

* 1. Способы создания форм (**🕮** Три способа создания)
  2. Автоматическое создание форм (кнопка Форма)
  3. Создание с помощью мастера
  4. Создание форм с помощью конструктора (**🕮** Создание БД и объектов и работа с ними \Объекты БД \Работа с формами \ Настройка форм \ Работа в режиме Конструктора)
     1. Окно приложения
     2. Структура формы (**🕮** **…**\Настройка формы \Работа с разделами форм)
     3. Панель инструментов *Конструктор форм*
     4. Панель инструментов *Формат*
     5. Параметры работы с формами
     6. Элементы управления и работа с ними (обратите внимание на изменение типа элемента управления)

1. Оформление формы и ее элементов
   1. Применение автоформата
   2. Изменение цвета фона формы и ее элементов
   3. Изменение фонового рисунка формы
   4. Форматирование элементов формы
   5. Копирование формата элемента управления
   6. Использование дополнительных цветов
2. Работа с записями в форме
   1. Добавление записи
   2. Изменение записи
   3. Отмена изменения записи
   4. Удаление записи
3. Настройка форм (**🕮** **…**\ Работа с формами \ Настройка форм)
   1. Отключение режима для формы (**🕮** Отключение)
   2. Разрешение или запрет устанавливать свойства формы (**🕮** Разрешение )
   3. Установка для формы атрибута «только для чтения» (**🕮**Установка )
   4. Использование стандартного формата (**🕮**Использование)
   5. Создание маски ввода (**🕮**Создание)

## РАЗРАБОТКА СЛОЖНЫХ ФОРМ

1. Изучение ЭЛЕМЕНТОВ УПРАВЛЕНИЯ (**🕮** Создание БД и объектов и работа с ними \ Работа с элементами управления и диаграммами»)
   1. Типы элементов управления (**🕮** Типы ЭУ)
   2. Панель элементов
   3. Разновидности элементов управления по типу содержимого
   4. Элемент управления Надпись
   5. Элемент управления Поле
   6. Элементы управления выбора: Выключатель, Переключатель, Флажок и Группа
   7. Элемент управления Список
   8. Элемент управления Поле со списком

Элемент управления Кнопка (**🕮** Создание командных кнопок в форме)

* 1. Элементы управления Свободная рамка объекта и Присоединенная рамка объекта
  2. Элемент управления Рисунок
  3. Элемент управления Разрыв страницы
  4. Элемент управления Набор вкладок
  5. Элемент управления Подчиненная форма/отчет
  6. Элементы управления Линия Прямоугольник
  7. Преобразование одного элемента управления в другой
  8. Использование диаграмм в формах (**🕮** Использование диаграмм в форме)
  9. Условное форматирование (**🕮** Условное форматирование)
  10. Работа с группами ЭУ (**🕮** ...\*Работа с группами ЭУ***)**

1. Создание многотабличных иерархических форм (**🕮** Создание БД и объектов и работа с ними \Объекты БД \Работа с формами\Создание подчиненных орм )
   1. Отображение подчиненной формы в главной форме. (**🕮** Отображение)
   2. Способы создания подчиненных форм (**🕮** Способы )
      1. Создание главной и подчиненной форм при помощи мастера автоформ
      2. Использование Мастера форм для создания главной и подчиненной форм
      3. Создание подчиненной формы и добавление ее в другую форму
      4. Создание формы с двумя подчиненными формами (главная форма содержит оба [элемента управления подчиненных форм](mk:@MSITStore:C:\Program%20Files\Microsoft%20Office\Office11\1049\acmain11.chm::/html/acdecCreateSubformS.htm))
      5. Создание формы с вложенными подчиненными формами (вторая подчиненная форма внедрена в первую)
2. Синхронизированные формы (🕮Синхронизированные формы )
3. Форма, состоящая из нескольких страниц или вкладок
   1. Форма, состоящая из нескольких страниц

***Примечание.*** В некоторых случаях может случиться так, что a форме должно содержаться больше информации, чем может быть выведено на экран. Или необходим, чтобы пользователь прошел определенную последовательность шагов при работе с формой. Многостраничные формы являются в этом случае подходящим решением. Лучшим наглядным примером многостраничной формы являются программы-мастера. Любая программа-мастер задает пользователю вопросы и затем создает объект, основываясь на полученных ответах.

* 1. Форма, состоящая из вкладок

1. Всплывающая форма и пользовательское диалоговое окно
   1. Всплывающая форма
   2. Пользовательское диалоговое окно. Разработайте диалоговое окно для запроса с параметром. Значение параметра выбирается из списка или поля со списком.
   3. Свойства всплывающих форм и отчетов
2. Использование вычислений в форме
   1. Создание вычисляемых полей
   2. Вычисление итогового значения в подчиненной форме и его вывод в форме
   3. Использование статистических функций в форме
3. Работа с элементами управления и диаграммами (**🕮**  Создание БД и объектов и работа с ними \ Работа с элементами управления и диаграммами»)
   1. Типы элементов управления
   2. Размещение командных кнопок в форме
   3. Использование диаграмм в форме
   4. Условное форматирование
4. Запрос по форме (query by form – QBF)
5. Просмотр сведений о зависимостях (Создание БД и объектов и работа ними \Просмотр сведений о зависимостях)

Изображение выглядит как текст

Автоматически созданное описание

## Основное задание

Сначала разработайте формы для ввода данных. При этом учитывайте структуру первичных документов. Грамотно установите правила перемещения по полям. Семантически связанные данные группируйте и графически выделите.

Как минимум одна форма должна быть создана в режиме конструктора. Кроме стандартных элементов управления в такую форму(ы) следует вставить дополнительные элементы управления. К наиболее полезным и часто используемым можно отнести ЭУ Календарь, Дерево (*TreeView*).

Обязательно наличие всех видов форм. Последние формы (диалоговое окно, кнопочная форма и формы сводных таблиц и диаграмм) разрабатываются позднее.

1. Разработайте формы для ввода данных. В процессе разработки технологии загрузки базы данных и проектирования форм целесообразно определить:

* перечень документов-источников, сохраняемых в базе и содержащих необходимые данные для загрузки таблиц базы данных;
* таблицы — объекты загрузки для каждого документа-источника;
* содержание и последовательность загрузки. При этом необходимо учитывать, что для обеспечения связной целостности главные таблицы должны быть загружены ранее подчиненных;
* подсхему данных каждой формы (фрагмент схемы данных), состоящую из таблиц, необходимых для создания электронного документа. При этом для многотабличной (составной) формы выбирается:
* таблица, которая будет базовым источником записей главной формы, и таблицы для отображения справочных данных в этой части формы;
* таблица, которая будет источником записей подчиненной формы, включаемой в главную форму, и таблицы для отображения справочных данных в подчиненной форме;
* макет формы, т. е. ее общую структуру, соответствующую структуре документа-источника и полученной подсхеме данных. При этом распределяется пространство формы для размещения включаемых подчиненных форм;
* состав и размещение элементов, связанных с полями таблиц, и надписей для каждой из частей составной формы. При этом:
* в главную форму обязательно надо вводить ключевые поля таблицы-источника данных (например, идентификатор документа «Договор» — номер договора);
* в подчиненной форме предусмотреть только те ключевые поля таблицы — базового источника подчиненной формы, которых нет в таблице-источнике главной формы (например, код товара из спецификации документа «Договор»).

После выполнения перечисленных пунктов и получения макета формы можно приступить к разработке форм средствами Access.

Ввод и корректировка справочных данных могут быть осуществлены через простые формы с макетом в столбец или табличный, в которых для проверки значений в полях заданы ограничения.

* ***Форму в столбец*** (кнопка Форма) используйте только в том случае, когда число полей достаточно велико и невозможно разместить их в одной строке. Предпочтение следует отдавать ***табличным формам***, как более наглядным.

Для ввода и корректировки данных плановых и оперативно-учетных документов пользователю нужно разработать удобный экранный интерфейс, который позволит минимизировать операции по вводу данных и контролировать их достоверность и корректность. При этом необходимо ограничиваться вводом только идентификаторов и количественных показателей. Справочные данные (наименования, нормативы, цены, тарифные ставки и т. п.) не могут вводиться с этих документов, а должны только отображаться в форме из ранее созданных таблиц справочной информации. Форма служит электронным документом, вид которого должен соответствовать виду бумажного документа.

1. Разработайте форму, основанную на запросе с параметрами.

 Пусть подформа основана на запросе с параметром (допустим, [Param]). Если в главной форме организовать элемент управления с именем [Param] (текстбокс, комбобокс, листбокс, т. е. элемент управления, имеющий свойство Value), тогда при открытии формы значение параметра для запроса-источника подформы будет браться из этого контрола [Param], соответственно, диалог «введите значения параметра» выводиться не будет, и, что важно, при изменении поля [Param] будет происходить автоматически *requery* подформы. (Кстати, это верно не только для подформ, можно использовать и некоторые другие объекты, например MsGraph).

1. В форме вычислите сумму или среднее значение для группы записей
2. Разработайте иерархические формы

 Иерархические формы удобно создавать при помощи мастера автоформ

* 1. Создайте подчиненную таблицу с подсчетом итоговых сумм по категориям (см. статью «[Итоги в формах](#_Итоги_в_формах)»)
  2. Для подчиненной формы задайте режим по умолчанию и отключите один или несколько других режимов. Для каждой разработанной иерархической формы [установите](mk:@MSITStore:C:\Program%20Files\Microsoft%20Office\Office11\1049\acmain11.chm::/html/achowSetOpeningViewFormSingleFormOneRecordContinuoS.htm) свой режим по умолчанию
  3. Создайте несколько иерархических форм с разными режимами отображения подчиненных форм. Режимы сводной таблицы и сводной диаграммы будут рассмотрены ниже в другой лабораторной работе.

1. Разработайте синхронизированные формы
2. Разработайте формы, состоящие из нескольких вкладок
3. Создайте запрос на основе форм (QBF – Query by Form). Данный подход позволяет пользователям предоставить доступ к сложным и весьма эффективным запросам, многочисленные параметры которых запоминаются с помощью полей формы. Значение параметра должны выбираться из списка (см. статью «[Запрос по форме](#_Запрос_по_форме)» из приложения).
4. Вставьте диаграмму в одну из разработанных форм. Совместите с работой диаграммы фильтр.
5. Создайте форму с использованием *Конструктора.* При этом используйте как можно большее число различных элементов управления.
   1. Покажите в форме использование не только области данных, но и областей заголовка, примечания и колонтитула
6. Разработайте формы с использованием *ЭУ Календарь* и *TreeView*.
7. Разработайте формы для отображения и выполнения запросов (на основе таблицы ***т\_РеестрЗапросов***, созданной в лабораторной работе *Запросы*).
8. Покажите применение условного форматирования в формах.
9. Для всех разработанных форм установите свойства:
   * Автоматический размер
   * Тип границы
   * Выравнивание по центру
   * Кнопки размеров окна
   * Допускается перемещение
   * Загрузить значения по умолчанию

## Содержание отчета

Все сведения о разработанных формах свести в единую таблицу



### Описание форм

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Название формы** | **Тип формы** | **Назначение** | **Примечание** |
| 1 | Учет заказов | Иерархическая | Анализ поступления и выполнения заказов | На форме размещены командные кнопки |
| 2 | Телефон | Простая | Справочник |  |
|  |  |  |  |  |
|  |  |  |  |  |

Привести распечатку двух самых сложных форм.



Для одной из формы привести список используемых свойств с их значениями (***Сервис/Анализ/Архивариус***), а также описание наиболее часто используемых свойств.



Приведите пример условного форматирования для одной из разработанных форм.



## 

## Контрольные вопросы

1. Какой самый быстрый способ создания формы?
2. В каких трех режимах форма может быть выведена на экран?  
   Как создать форму вручную?
3. Как добавить в форму поле текущей даты или времени?
4. Как добавить в форму список или поле со списком?
5. Как предотвратить запуск мастера при создании кнопки, списка или поля со списком? ( в режиме конструктора формы или отчета отключить *Мастера* на ПИ)
6. Пусть есть список, состоящий из двух или более столбцов. Как сделать так, чтобы один столбец был скрыт и не отображался?

 Такой прием может быть полезен, если у вас есть список, присоединенный к полю, а содержимое этого поля вы не хотите отображать.

1. Как создать подчиненные формы до семи уровней вложенности?
2. Дайте понятие «присоединенного элемента управления»
3. Как можно использовать поле со списком для поиска записей в форме?
4. Как создать всплывающую подсказку, когда указатель мыши подводится к какому-либо элементу управления?
5. Как сделать, чтобы числа в форме отображались в денежном формате?
6. Как сохранить выравнивание элементов управления при перемещении по горизонтали или вертикали? (удерживать нажатой Shift) 1
7. Можно ли использовать собственный рисунок в качестве фона?
8. Как вручную добавить ЭУ в форму? (самый простой способ – это воспользоваться списком полей)
9. Как преобразовать один ЭУ в другой?
10. Как разыскать запись по ее номеру, не выходя из режима формы?
11. Как выполнить фильтрацию отображаемых записей во время работы с формой?
12. Как добавить в форму кнопку для печати одной текущей записи?
13. Как создать несколько подчиненных форм в одной форме?
14. Как сделать подчиненную форму доступной только для чтения и при этом разрешить внесение изменений в данные главной формы?
15. Как вычислить итоговые значения в подчиненной форме и отобразить результаты в главной форме? Что такое «область выделения формы», «[область выделения раздела](mk:@MSITStore:C:\Program%20Files\Microsoft%20Office\Office11\1049\acmain11.chm::/html/achowSelectSectionFormReportS.htm)»?
16. Где используется условное форматирование?
17. Как сделать подсчет положительных и отрицательных значений, содержащихся в логическом поле?
18. Как отобразить в форме или отчете информацию из другой формы?

Это можно сделать с помощью функции [Dlookup(),](mk:@MSITStore:C:\Program%20Files\Microsoft%20Office\Office10\1049\vbaac10.chm::/html/acmthdlookup.htm) которая возвращает значение поля.

1. Как просмотреть базовый запрос для отчета или формы?
2. Можно ли введенное в текстовое поле значение использовать в качестве фильтра?
3. Как в запросе с параметром вводить не все значение текстового поля , а только его часть. Например, при поиске записей о человеке пользователь может ввести только часть полного имени, в то время как запрос должен разыскать все записи, в которые входит введенный фрагмент. Как это сделать?
4. Как организовать синхронизацию главной и подчиненной форм?

## Дополнительное задание

Форма, состоящая из нескольких страниц.



## ПРИЛОЖЕНИЕ

##### Содержание

* [Типы форм](../../../Users/PC/AppData/Roaming/Microsoft/Word/Типы%20форм.doc)
* [Как правильно ссылаться на элементы управления в формах](#_Как_правильно_ссылаться_на элементы)
* [Нумерация строк запроса](#_Нумерация_строк_запроса_(http://hip)
* [Суммирование с накоплением в запросе](#_Суммирование_с_накоплением_в запрос)
* [Запросы с полями из форм](#_Запросы_с_полями_из форм (О присвое)
* [Запрос по форме](#_Запрос_по_форме)
* [Итоги в формах](#_Итоги_в_формах)
* [Элемент TreeView в Visual Basic](#_Элемент_TreeView_в_Visual Basic)
* [Создание всплывающей формы](#_Создание_всплывающей_формы_или отче)
* Создание масок ввода

## Типы форм

Чтобы удовлетворить разнообразным требованиям, в Microsoft Access можно создавать различные типы форм.

Форма, основанная на нескольких таблицах или запросах

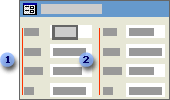


Использование мастера форм является самым удобным и быстрым способом создания формы, в которой объединены сведения из разных таблиц или запросов. При использовании мастера форм для выбора полей из одной или нескольких таблиц или запросов в приложении Microsoft Access создается простая или иерархическая форма, а также инструкция SQL для связывания формы с источником записей. Инструкция SQL содержит сведения об используемых таблицах, запросах и полях.

Простая форма



Простая форма является обычной формой, содержащей элементы управления, связанные с различными таблицами и запросами. Примером такой формы может служить форма, содержащая сведения о товарах и поставщиках товаров.



В эти поля данные подставляются из таблицы «Товары».



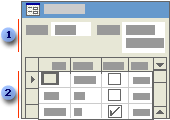
В эти поля данные подставляются из таблицы «Поставщики».



Иерархическая форма



Иерархической называется форма, содержащая одну или несколько внедренных форм. Внедренной формой называется подчиненная форма. Подчиненные формы удобны для отображения данных из таблиц, связанных отношением «один-ко-многим». Например, можно создать форму «Типы», содержащую данные из таблицы «Типы», а также данные из таблицы «Товары».



Данные этих полей взяты из таблицы «Типы» (на стороне «один» отношения «один-ко-многим»).



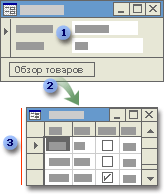
Данные этих полей взяты из таблицы «Товары» (на стороне «много» отношения «один-ко-многим»).



Синхронизированные формы



Может возникнуть необходимость представить данные в иерархическом виде без использования подчиненных форм. Например, если в форме много элементов управления, для подчиненной формы может не остаться места. В этом случае можно использовать мастер форм для создания синхронизированных форм. При нажатии кнопки в форме открывается другая форма, синхронизированная с записью в первой форме.



Эта форма содержит данные из таблицы «Поставщики» (на стороне «один» отношения «один-ко-многим»).



Эта кнопка



отображает связанные записи из таблицы «Товары» (на стороне «многие» отношения «один-ко-многим»).



Форма, состоящая из нескольких страниц или вкладок

Форма, состоящая из нескольких страниц

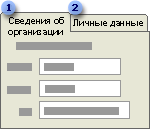


Для создания формы, содержащей несколько страниц, можно использовать элемент управления «Разрыв страницы». Элемент управления «Разрыв страницы» используется для указания горизонтальных разрывов между элементами управления в форме. Для перехода к странице, находящейся над или под указанным разрывом, используются клавиши PAGE UP или PAGE DOWN.

Форма, состоящая из вкладок



Для создания формы, содержащей вкладки, можно использовать элемент управления «Набор вкладок». Использование элемента управления «Набор вкладок» при создании формы, состоящей из нескольких страниц, является самым простым и эффективным способом.



Общие сведения о сотруднике отображаются на этой вкладке.



Личные данные, такие как адрес и домашний телефон, отображаются на второй вкладке.



Всплывающая форма и пользовательское диалоговое окно

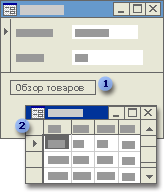


Всплывающая форма служит для вывода сообщений для пользователя или для ввода данных пользователем. Всплывающая форма располагается поверх других открытых форм, даже если активной является другая форма. Всплывающая форма может быть немодальной или модальной. Модальные всплывающие формы называют пользовательскими диалоговыми окнами.

Всплывающая форма



При открытии немодальной всплывающей формы пользователь имеет доступ к другим объектам и командам меню. Например, в форму «Поставщики» можно добавить кнопку, выводящую всплывающую форму «Список товаров». Эта всплывающая форма отображает список товаров, доступных через текущего поставщика в форме «Поставщики».



При нажатии этой кнопки открывается немодальная всплывающая форма.



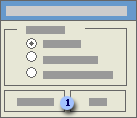
Всплывающая форма располагается поверх других открытых форм, однако можно перевести фокус в другое окно, не закрывая при этом форму.



Пользовательское диалоговое окно



Если всплывающая форма является модальной, доступ к другим объектам или командам меню невозможен до скрытия или закрытия этой формы. Например, можно создать пользовательское диалоговое окно для выбора нужного отчета для печати.



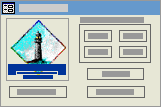
Пользовательское диалоговое окно располагается поверх других окон. Для перевода фокуса в другое окно необходимо закрыть или скрыть форму.



Кнопочная форма



При создании базы данных с помощью мастера автоматически создается кнопочная форма, помогающая перемещаться по базе данных. На кнопочную панель помещаются кнопки, при нажатии которых открываются формы или отчеты (или открываются другие кнопочные формы, с помощью которых открываются дополнительные формы и отчеты), осуществляется выход из программы Microsoft Access или настраивается сама кнопочная форма. Пользователь может создать кнопочную форму, аналогичную формам, создаваемым при помощи мастера баз данных с использованием диспетчера кнопочных форм.

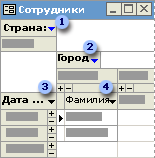


Форма в режиме сводной таблицы или сводной диаграммы



Присоединенную форму можно открыть в режиме сводной таблицы или сводной диаграммы. В этих режимах можно добавлять или перемещать поля в различные области представления, чтобы изменить макет или внешний вид данных. Можно также применять фильтры и выполнять сортировку и группировку данных в представлении.

Форма в режиме сводной таблицы



Поле фильтра



Поле столбца



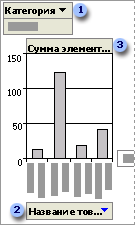
Поле строки



Поле сведений



Форма в режиме сводной диаграммы



Поле фильтра



Поле типа



Поле данных



## Как правильно ссылаться на элементы управления в формах

В принципе эта тема довольно подробно рассмотрена в Help'e, но почему-то для многоих этот вопрос по-прежнему остается тайной.

Маленький совет: имя элемента управления никогда не должно совпадать с данными этого элемента управления:

Как на этом скриншоте: поле имеет ***имя*** **ctlDs\_City** и содержит ***данные*** **Ds\_City**. При этом ссылка **Me!ctlDs\_City** однозначно указывает на ***элемент управления***, а ссылка **Me!Ds\_City** - на ***данные источника записей формы***. Например, если бы в этом случае ***имя*** и ***данные*** совпадали, то, например, поле в примечании с данными «**=Count([Ds\_City])**» выдало бы ошибку.

**Ссылки на формы:**

Пусть у нас есть форма с именем frm\_MainForm. Обратиться к ней можно несколькими способами:

|  |  |
| --- | --- |
| **CurrentDb.Containers(1).Documents(“frm\_MainForm”)** | Откуда угодно когда угодно |
| **Forms!frm\_MainForm Forms![frm\_MainForm] Forms(“frm\_MainForm”) Forms.Item(“frm\_MainForm”)** | Откуда угодно если форма frm\_MainForm открыта. Всесто «frm\_MainForm» можно подставлять переменную, содержащую имя формы. |
| **Me** | Из формы frm\_MainForm |
| **Screen.ActiveForm** | Откуда угодно если форма frm\_MainForm в данный момент времени является активной формой на экране |
| Также возможны и другие способы ссылки на форму - по индексу в семействе Forms, по индексу Documents в Containers(1) и т.д. | |

В дальнейшем для краткости будем обращатся к элементам управления формы из самой формы через Me.

**Ссылки на элементы управления формы:**

Пусть имеем в форме несколько элементов управления: поле **ctlField**, надпись **ctlText**, список **ctlList**, набор вкладок **ctlPages** с одной вкладкой с именем **ctlPage01** на которой есть прямоугольник **ctlRect**, подформу **ctlSubForm** в которой находить форма с именем **frm\_SubForm**, в которой в свою очередь есть кнопка **ctlButton**.

|  |  |
| --- | --- |
| **Me!ctlField Me![ctlField] Me.Controls(“ctlField”) и т.д.** | Ссылка на поле ctlField |
| **Me!ctlText.Caption и т.д.** | Ссылка на свойство «Подпись» надписи ctlText |
| **Me!ctlList.Column(0) и т.д.** | Ссылка на 1-й столбец списка ctlList |
| **Me!ctlPages и т.д.** | Ссылка на набор вкладок ctlPages |
| **Me!ctlPages.Pages(“ctlPage01”) Me!ctlPages.Pages(0) Me!ctlPages01 и т.д.** | Ссылки на 1-ю вкладку (ctlPage01) из набора вкладок ctlPages |
| **Me!ctlPages.Pages(“ctlPage01”).Controls(“ctlRest”).BackColor Me!ctlPages.Pages(0).Controls(“ctlRest”).BackColor Me!ctlPages01.Controls(“ctlRest”).BackColor Me!ctlRest.BackColor и т.д.** | Ссылки на свойство «Цвет фона» прямоугольника ctlRest |
| **Me!ctlSubForm.SourceObject и т.д.** | Ссылка на свойство «Объект-источник» элемента управления ctlSubForm |
| **Me!ctlSubForm.Form.RecordSource и т.д.** | Ссылка на свойство «Источник записей» подформы |
| **Me!ctlSubForm.Form!ctlButton.Enabled и т.д.** | Ссылка на свойство «Доступ»  кнопки ctlButton в подформе. |
| **Me.Parent!ctlField и т.д.** | Ссылка на поле ctlField основной формы из подформы. |

## Нумерация строк запроса

## (<http://hiprog.com/access/article.asp?id=359>)

**Автор:** [**Юрий Шерман**](mailto:sherman-ruprog@mtu-net.ru)[**www.tour-soft.com**](http://www.tour-soft.com/)

О старом подходе.

Вопрос о нумерации строк запроса неоднократно задавался на форуме.  
Конечным ответом всегда была ссылка:  
<http://am.rusimport.ru/MsAccess/topic.aspx?ID=87>, где статья дается в разделе «Содержание. Запросы. Порядковый номер записи»  
Сама статья имеет ссылку на источник: [http://www.dykbits.com](http://www.dykbits.com/)

В статье приводятся три метода нумерации со следующей характеристикой, указанной автором:  
»Первые два - работают существенно быстрее последнего, но последний запрос обладает одним очень важным достоинством - данные, полученные при помощи этого запроса, можно модифицировать.»   
То есть, первые два метода предлагают статические запросы, третий - динамический.

По текстам запросов нетрудно дать общую оценку скорости исполнения запросов: C\*n\*n, где С - константа, различная для запросов; n - количество строк в исходной таблице (запросе). Требование выходной сортированности только ухудшает эту оценку.

Однако автором пропущен важный недостаток всех методов: значения поля, на основе которых производится нумерация, не должны дублироваться.

Пример. Если Вы имеете исходную таблицу с одним полем и пятью строками с одним значением, то в поле Номер Вы получите везде цифру 5. То есть, задача не решена - нумерация не выполнена.

Более мелкий недостаток: во всех методах используется сравнение значений разных строк поля. Однако, как известно, сравнение можно проводить не всех типов полей.

Новый подход для получения статических запросов.

Предлагается метод нумерации с оценкой скорости C\*n, не зависящий от значений и типов исходных данных.  
Очевидно, что эти характеристики уже улучшить нельзя. То есть метод оптимален.

Постановка задачи 1. Запрос без полей MEMO и OLE.

Пусть имеется таблица (запрос) MyQuery с полем MyField. В запросе могут быть другие поля, допускающие предикат DISTINCT. MyQuery может быть отсортирован любым образом или вообще не отсортирован.

Требуется построить запрос, не меняющий порядка следования записей MyQuery, но добавляющий вычисляемое поле Num, представляющее номер строки MyQuery, начиная с 1.

Решение.

Текст запроса:

SELECT DISTINCT Numeration(MyField) As Num, MyField FROM MyQuery WHERE Numeration() = 0;

В запросе используется функция Numeration. Вот ее описание:

Public Function Numeration(Optional var) As Long  
Static n As Long  
If IsMissing(var) Then  
n = 0  
Else  
n = n + 1  
End If  
Numeration = n  
End Function

**Комментарий 1.** Запрос оптимизируется компилятором SQL. Поэтому значение функции в предложении WHERE, не зависящее от поля запроса , вычисляется только один раз. Этот вызов используется для установления начального значения счетчика.

**Комментарий 2.** Предикат DISTINCT используется для превращения динамического запроса в статический. (В динамическом запросе метод не работает.) Своей прямой роли предикат DISTINCT не играет, так как все строки результирующего запроса уникальны из-за нумерации. Однако он препятствует использованию в запросе полей MEMO и OLE.   
Эти поля можно преобразовать в тип String, применяя функцию Mid(Field\_MEMO\_OLE,1). После чего конфликт исчезнет, но поля OLE потеряют свою функциональность. Если есть только поля MEMO, то такой подход применим.

Постановка задачи 2. В запросе допускаются любые поля.

Пусть имеется таблица (запрос) MyQuery с полем MyField. В запросе могут быть любые другие поля. MyQuery может быть отсортирован любым образом или вообще не отсортирован.

Требуется построить запрос, не меняющий порядка следования записей MyQuery, но добавляющий вычисляемое поле Num, представляющее номер строки MyQuery, начиная с 1.

Решение.

Текст запроса:

SELECT Numeration(MyField)/2 As Num, First(MyField) AS MyFieldFirst   
FROM MyQuery   
WHERE Numeration() = 0   
GROUP BY Numeration(MyField)/2;

**Комментарий.** Здесь запрос превращается в статический за счет применения группировки. Это снимает все требования с полей, но усложняет запрос и увеличивает время его исполнения. Для полей MEMO и OLE следует применять агрегатную функцию First или Last, для остальных достаточно указания группировки. Реально группировка также не проводится, так как первый уровень группировки (Numeration(MyField)/2)всегда дает одиночные записи.

О нумерованном динамическом запросе.

Улучшить метод получения нумерованного динамического запроса не удалось. Более того, разработать метод, нумерующий запрос со скоростью = C\*n, по-видимому, невозможно. Поэтому для полноты привожу метод 3 из упомянутой выше статьи.

SELECT DCount(«\*», «MyQuery»,КритерийСравнения) AS Num, MyField  
FROM MyQuery ORDER BY MyField;

КритерийСравнения для чисел: «MyField <=» & MyField

КритерийСравнения для текста: «MyField <='» & MyField & «'»

Примечание.

Следует отметить, что реальная необходимость применения нумерованного динамического запроса возникает в только формах, предусматривающих корректировку данных. А в этом случае можно применить следующий прием:

* форму строим на нумерованном статическом запросе;
* корректировку проводим в свободном поле формы;
* после корректировки изменяем прямым доступом соответствующее поле в основной таблице;
* обновляем запрос в форме.

Примерно таким же образом можно ввести и новую запись. Все это, конечно, осложняет программирование, но резко увеличивает скорость исполнения. Разница в скоростях исполнения отчетливо видна уже на запросах в 1000 записей.

## Суммирование с накоплением в запросе

**Автор:** [**Юрий Шерман**](mailto:sherman-ruprog@mtu-net.ru)[**www.tour-soft.com**](http://www.tour-soft.com/)

Запрос, нумерующий записи и выдающий сумму полей с накоплением.

Постановка задачи.

Пусть имеется таблица (запрос) MyQuery с числовым полем MyField. Другие поля также могут быть, но далее в тексте не присутствуют. MyQuery может быть отсортирован любым образом или вообще не отсортирован.

Требуется построить статический запрос, не меняющий порядка следования записей MyQuery, и добавляющий:  
- вычисляемое поле Num, нумерующее строки запроса, начиная с 1;  
- вычисляемое поле Summa, представляющее собой сумму значений поля MyField от первой строки до текущей.

Решение.

Текст запроса:

SELECT DISTINCT Numeration(MyField) as Num, SumField(MyField) As Summa, MyField FROM MyQuery  
WHERE Numeration() = 0 AND SumField() = 0;  
  
Функция Numeration описана в статье [“Нумерация строк запроса”](http://hiprog.com/access/article.asp?id=359)

Описание функции SumField:  
  
Public Function SumField(Optional Var) As Double  
Static D As Double  
If IsMissing(var) Then  
    D = 0  
Else  
    D = D + Nz(Var,0)  
End If  
SumField = D  
End Function

Оценка скорости исполнения.

Очевидно, что обращение к каждой строке исходного запроса производится только один раз. Поэтому оценка скорости  = C\*n, где C - константа, а n - число строк запроса. Качественно ускорить это решение невозможно.

Примечание.

Если в результирующем запросе все строки уникальны (например, присутствует поле «Счетчик»), то использовать нумерацию нет необходимости. Она применяется только для создания уникальных записей.

О динамических запросах, решающих аналогичную задачу.

Можно построить динамический запрос, решающий эту задачу. Однако на исходные данные придется наложить существенные ограничения. Но самое главное, такой запрос будет исполняться со скоростью = C\*n\*n. Избежать этого, видимо, невозможно.

Следует и здесь применить метод, изложенный в примечании к вышеупомянутой статье, который только изображает работу с динамическим запросом, реально используя статический.

## Запросы с полями из форм (О присвоении параметров при открытии Recordset)

Автор: [**Юрий Шерман**](mailto:sherman-ruprog@mtu-net.ru)

[**www.tour-soft.com**](http://www.tour-soft.com/)

О присвоении параметров при открытии Recordset

Часто используются запросы, включающие значения полей форм. Обычно их включают в условия отбора.

Однако, если Вы попробуете открыть Recordset на основе такого запроса, то Access даст сообщение об ошибке: «Указаны не все параметры ...». Эти параметры как раз поля из форм.

Ниже приводится простейшая функция, подставляющая значения параметров (полей форм) в запрос.

‘Открытие Recordset на базе запроса с параметрами   
’в виде полей форм  
Public Function OpenRQ(NameQuery As String) As Recordset  
Dim Q As QueryDef, par As Parameter  
Dim S As String, i As Integer, j As Integer  
  
Set Q = CurrentDb.QueryDefs(NameQuery)  
  
For Each par In Q.Parameters  
   S=par.Name  
   i = InStr(S, “!”)  
   j = InStr(i + 1, S, “!”)  
   Par = \_  
Forms(Mid(S, i + 1, j - i - 1))(Mid(S, j + 1))  
Next par  
  
Set OpenRQ = Q.OpenRecordset  
  
End Function



Следующая функция немного сложнее, но она делает то же самое не только с запросами,   
записанными в базу (QueryDefs), но и запросами, заданными прямым текстом SQL.  
Конечно, SQL-текст может содержать ссылки на другие запросы.

Public Function rpOpenRQ(NameQuery\_Or\_SQL As String) \_  
As Recordset  
Dim Q As QueryDef, par As Parameter  
  
'В тексте SQL обязательно есть FROM c пробелами  
If NameQuery\_Or\_SQL Like «\* FROM \*» Then  
'текст SQL - создаем временный запрос  
   Set Q = \_  
CurrentDb.CreateQueryDef(«», NameQuery\_Or\_SQL)  
  
Else  
'имя запроса  
   Set Q = CurrentDb.QueryDefs(NameQuery\_Or\_SQL)  
  
End If  
  
For Each par In Q.Parameters  
   par = rpGetVal(par.Name)  
Next  
  
Set rpOpenRQ = Q.OpenRecordset  
End Function



Здесь используется функция rpGetVal. Она выдает значение поля формы, заданное текстом: «[Forms]FormName]![FieldName]». (Квадратные скобки могут отсутствовать.) Кроме того, она выдает числовое значение, заданное в виде «=2», и текст без преобразования.

Эти возможности нужны для восстановления значений полей форм, заданных по умолчанию.

Например, FieldName.DefaultValue=”=Forms!A!B” или FieldName.DefaultValue=”=5”.

Для восстановления начального значения поля достаточно написать:

FieldName=rpGetVal(FieldName.DefaultValue)

Прямое присвоение: FieldName=FieldName.DefaultValue обычно не проходит.

Public Function rpGetVal(Expression)  
Dim S As String, i As Long, j As Long, d As Double  
  
’Возвращаем Null при пустых значениях Expression.  
If IsNull(Expression) Then Exit Function  
S = Expression  
If S = “” Then Exit Function  
  
’Убираем знак “=” - он может быть в DefaultValue.  
If Left(S, 1) = “=” Then S = Mid(S, 2)  
  
’Разбор значения S  
On Error GoTo EndFun   
’Если ошибка при разборе, то выход = Null  
i = InStr(S, “!”)  
If i > 0 Then  
’Это поле формы  
   j = InStr(i + 1, S, “!”)  
   rpGetVal = \_  
Forms(Mid(S, i + 1, j - i - 1))(Mid(S, j + 1))  
Else  
’Может это число?  
    If IsNumeric(S) Then  
’Это число. Дробное?  
       d = CDbl(S)  
       i = CLng(S)  
        If d = i Then  
           rpGetVal = i ‘целое  
        Else  
           rpGetVal = d ‘дробное  
        End If  
    Else  
’Это какой-то текст  
       rpGetVal = S  
    End If  
End If  
Exit Function  
  
EndFun:  
End Function



## Запрос по форме

Данный подход позволяет пользователям предоставить доступ к сложным и весьма эффективным запросам, многочисленные параметры которых запоминаются с помощью полей формы.

Простейшим запросом QBF моет быть форма, на которой представлены все поля заполняемой таблицы. Пользователь просто вводит данные в поля и щелкает на кнопке *Поиск* или *OK* Щелчок на кнопке запускает запрос, возвращающий из таблицы записи со значениями, совпадающими с введенными в поля формы.

***Можно ли ввести что-либо в текстовое поле формы и использовать введен­ное значение в качестве фильтра?***

Для этого можно использовать метод, который называется запрос по форме (query-by-form). В строку **Условие отбора** (*Criteria*) бланка запроса по образцу (QBE) вводится выражение, которое относится к текстовому полю формы, после чего к форме добавляется кнопка, щелчок на которой является командой для выполнения запроса. Например, на рис. 1.1 показана форма, используемая для ввода условия отбора, и результирующий набор записей, получаемый после щелчка на кнопке **Найти** (*Search*).

![Изображение выглядит как стол

Автоматически созданное описание

Рис. 1 1. Пример поиска по условию, введенному в форме

Создание запроса по форме выполняется следующим образом.

1. Создайте форму с текстовыми полями, на основе которых в запросе будет формироваться желаемое условие отбора. Вам понадобится по одному тексто­вому полю на каждую из ячеек бланка запроса по образцу, для которых предусмотрено условие отбора. Например, если вы хотите, чтобы поля запроса Город (City) и Штат (State) заполнялись в соответствии с информацией, взятой из формы, в форму необходимо добавить два текстовых поля.
2. На Панели элементов (Toolbox) включите кнопку-переключатель Мастера (Control Wizards), если она еще не включена.
3. Выберите элемент управления Кнопка (Command button) и добавьте его к форме. Откроется диалоговое окно мастера Создание кнопок (Control Button Wizard). В списке Категории: (Categories) выберите пункт Разное (Miscellaneous). В списке Действие (Actions) выберите пункт Запуск запроса (Run Query) и щелкните на кнопке Далее (Next).
4. В очередном диалоговом окне выберите запрос, который будет принимать и обрабатывать значения, введенные в форму. Щелкните на кнопке Далее.
5. В следующем диалоговом окне включите переключатель Текст (Text) и введите в текстовое поле надпись, которая должна быть изображена на кнопке, например: Найти. Щелкните на кнопке Готово (Finish) ­- кнопка будет добавлена в форму.
6. Сохраните форму.
7. Откройте запрос в режиме конструктора. В каждом текстовом поле строки Условие отбора: (Criteria), соответствующем текстовым полям формы, необходимо ввести выражение, например:

***Forms![имя формы] ! [имя элемента управления] OR Forms! [имя формы] ! [имя элемента управления] Is Null***

Здесь *[имя формы]* — имя вашей формы, а *[имя элемента управления] -*имя, присвоенное элементу управления «текстовое поле». Например, если форма называется *Поиск клиента*, то в строке Условие отбора: (Criteria) поля *Город* в запросе должна быть следующая запись:

*[Forms]![Поиск клиента]![Город] OR [Forms]![Поиск клиента]![Город] Is Null*

Сохранив запрос, можно открыть форму, ввести условие отбора в текстовые поля и щелкнуть на кнопке, чтобы выполнить запрос с введенными условиями отбора.



Смотрите видео в Интернете по ссылке

[Запрос с выпадающим списком в базе данных Access - поиск Яндекса по видео (yandex.ru)](https://yandex.ru/video/preview/15890004210754465216)

## Итоги в формах

##### Суммовые поля в форме

Если у вас в форме достаточно большое количество вычисляемых полей (промежуточные суммы, итоговые значения, справочные данные из других таблиц) то можно для них создать подчиненную форм, для которой источником данных может быть процедура, выдающая название суммы и ее значение –получается очень просто и красиво.

Относительно ошибки в итоговых полях, получаемых встроенной функцией =Sum()**, м**ожно посмотреть пример:

* Создаем форму MYFORM
* Создаем на ней элемент управления MYLISTBOX - типа LISTBOX
* Источником данных для этого поля конструируем запрос, вычисляющий суммы по группам
* Создаем кнопку, по нажатию на которую выполняется тестовый код
* Делаем событие «Click» для кнопки и пишем следующий код, который обновляет набор записей.

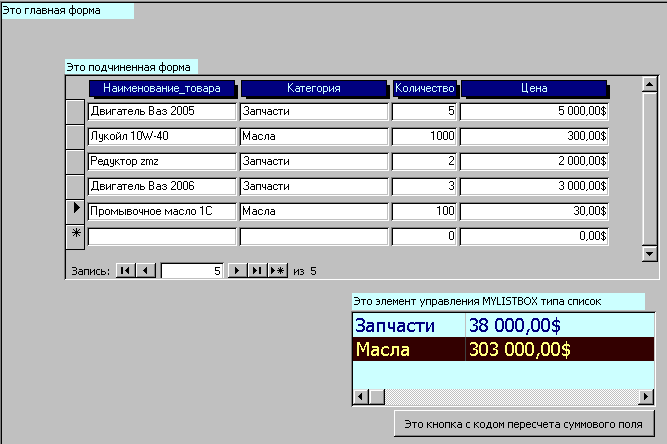
Public Sub MYBUTTON\_Click()

Me.MYLISTBOX.RowSource = Me.MYLISTBOX.RowSource

End Sub

Выглядит это так:

Пример можно скачать [здесь](http://kozin1.narod.ru/down_files/sum_ctrl.mdb)   
  
Относительно ошибки в итоговых полях, получаемых встроенной функцией =Sum()



Конец формы

## Элемент TreeView в Visual Basic

Итак, начнем с того, что такое TreeView?

Откройте Explorer (если у вас, конечно, Windows :)) — слева, как вы догадались, TreeView, а справа — ListView. Элемент TreeView обеспечивает средства хранения иерархически структурированных данных. Для хранения рисунков (таких, как папка, флаг Виндов для неопределенного типа файлов и т.п.) используется элемент ImageList — работа с ним довольно проста, и касаться этой темы пока не будем… Откроем новую форму и создадим элемент TreeView, который находится в контроле MS Windows Common Controls. Создадим также CommandButton, в событии Click которого мы и будем всё описывать.

Для начала теория. В этом элементе реализована структура данных, известная как «дерево», а каждый элемент дерева называется узлом (node). Верхний узел дерева называют корневым узлом (root node), а узлы ниже — дочерними узлами (child nodes). Рассмотрим это на структуре Интернета: домен ru — корневой узел, а домен narod для ru — дочерний узел. Домен narod для домена soobcha-vb является родительским. С этим, я думаю, всё понятно.

Теперь посмотрим свойства элемента TreeView: чтобы он выглядел как проводник, надо поставить значение Style равное 7, а LineStyle — 0. Там ещё много свойств, и я думаю, все их описывать не нужно — многие из них стандартные.

Теперь перейдем в окно кода и наберем TreeView1.Nodes, а дальше поставим точку. Увидели: в семейство Nodes входят 5 компонентов, с которыминам и предстоит работать. Начнем с основного и наиболее сложного: Add.

Создадим корневой узел:

TreeView1.Nodes.Add , , “internet”, “Internet”

Запустим программу и нажмем на кнопку — увидели? — да, появился узел Интернет. Но пока ничего не понятно — зачем столько запятых и два «Интернета». Давайте продолжим создавать узлы, и всё станет понятно! Введем:

TreeView1.Nodes.Add «Internet», tvwChild, «ru», «ru»

Ввели, а теперь нажимаем на старт и на кнопку, и если вы всё аккуратно ввели, или скопировали отсюда, то появилась ошибка. Начинаем разбиратся: что же мы хотели сделать? Создать дочерний узел ru к корневому Internet — на это указывает значение tvwChild — я думаю, оно понятно. Но откуда ошибка? После этого мы всё ввели правильно, ведь первая строка работала. Осталось одно значение — Internet. Ну, догадались? Именно, в первой строке, если вы внимательно следили за работой программы, в TreeView появилось Internet (с большой буквы), а internet (с маленькой) — это всего лишь ключ. И во второй строке мы указываем не прямое имя, а ключ этого имени, т.е. internet. Исправим, запустим, щелкнем двойным кликом на Internet — и вот он, дочерний узел ru. Создадим, соответственно, ещё пару доменов 1 уровня:

TreeView1.Nodes.Add «internet», tvwChild, «com», «com»   
TreeView1.Nodes.Add «internet», tvwChild, «net», «net»   
TreeView1.Nodes.Add «internet», tvwChild, «edu», «edu»

Вам не надоело всё время щелкать по Internet, да? — ну, введем:

TreeView1.Nodes.Item(“internet”).Expanded = True

Сложно — а кто говорил, что просто? Вот мы частично познакомились с компонентом Item. Ну, добавим ещё пару дочерних узлов к ru, тут уже всё понятно — вы и сами можете сделать это.

TreeView1.Nodes.Add «ru», tvwChild, «narod», «narod»   
'следуя от дочернего узла к родительскому, вы попадете на сайт Народ.ру   
TreeView1.Nodes.Add «narod», tvwChild, «www», «www»   
'а следуя отсюда, попадете на сайт, откуда вы это читаете   
TreeView1.Nodes.Add «narod», tvwChild, «soobcha-vb», «soobcha-vb»   
TreeView1.Nodes.Add «ru», tvwChild, «boom», «boom»   
TreeView1.Nodes.Add «ru», tvwChild, «lgg», «lgg»

Теперь сделаем путь к другому сайту бесплатного хостинга — [www.boom.ru](http://www.boom.ru) Все, наверное, введут

TreeView1.Nodes.Add “boom”, tvwChild, “www”, “www”

И снова ошибка! Просто вы уже привыкли, что ключ совпадает (во всяком случае, по значению — помните историю с регистром?). Но ведь не может быть два одинаковых ключа — как тогда добавлять дочерние узлы? В связи с этим необходимо пользоваться «полным» путем к узлу, т.е. для ru — это было бы internet-ru, для [www.narod.ru](http://www.narod.ru) — internet-ru-narod-www. Тут совпадений быть не может. Достоинства и того, и другого варианта вы поймете со временем: если вы уверены, что будут отсутствовать два узла с одинаковыми именами (хотя такое бывает редко), то имеет смысл оставить первоначальный вариант, а любом другом случае необходимо использовать «полный» путь. Получили вот такой код:

TreeView1.Nodes.Add , , “internet”, “Internet”   
TreeView1.Nodes.Add “internet”, tvwChild, “internet-ru”, “ru”   
TreeView1.Nodes.Add “internet”, tvwChild, “internet-com”, “com”   
TreeView1.Nodes.Add “internet”, tvwChild, “internet-net”, “net”   
TreeView1.Nodes.Add “internet”, tvwChild, “internet-edu”, “edu”   
TreeView1.Nodes.Item(“internet”).Expanded = True   
TreeView1.Nodes.Add “internet-ru”, tvwChild, “internet-ru-narod”, “narod”   
TreeView1.Nodes.Add “internet-ru-narod”, tvwChild, “internet-ru-narod-www”, “www”   
TreeView1.Nodes.Add “internet-ru-narod”, tvwChild, “internet-ru-narod-soobcha-vb” \_   
, “soobcha-vb”   
TreeView1.Nodes.Add “internet-ru”, tvwChild, “internet-ru-boom”, “boom”   
TreeView1.Nodes.Add “internet-ru-boom”, tvwChild, “internet-ru-boom-www”, “www”   
TreeView1.Nodes.Add “internet-ru”, tvwChild, “internet-ru-lgg”, “lgg”   
TreeView1.Nodes.Add “internet-ru-lgg”, tvwChild, “internet-ru-lgg-www”, “www”

Сложновато, на первый взгляд, но ведь мы получили такую сложную структуру… Теперь поговорим об остальных компонентах Nodes: Count — возвращает число узлов семейства Nodes. Введем:

MsgBox TreeView1.Nodes.Count

У нас появится в окошке количество узлов — в данном случае 12. Вы случайно ни разу не нажимали на вашу кнопку ещё раз? — правильно, появляется ошибка, т.к. ключи уже все созданы, и такие значения уже имеются. Поставьте в начале:

TreeView1.Nodes.Clear

И перед добавлением новых значений TreeView будет полностью очищаться. А если в процессе работы программы надо удалить узел? — правильно, надо воспользоваться командой Remove:

TreeView1.Nodes.Remove (“internet-ru”)

Удаляем узел ru, который мы так долго создавали.

А теперь немного о картинках:

TreeView1.Nodes.Add “internet”, tvwChild, “internet-org”, “org”, \_   
App.Path & “\pic1.bmp”, App.Path & “\pic1sel.bmp”

Таким образом показывается картинка, находящаяся в папке, где у вас сохранена эта программа (или вы её ещё не сохранили?), с именем pic1.bmp, и если узел выбран, то с именем pic1sel.bmp. Просто, да? А вот то же самое, только с использованием ImageList (он находится в MS Windows Common Controls-2):

ImageList1.ListImages.Add , , LoadPicture(App.Path & “\pic1.bmp”)   
ImageList1.ListImages.Add , , LoadPicture(App.Path & “\pic1sel.bmp”)   
TreeView1.ImageList = ImageList1   
TreeView1.Nodes.Add “internet”, tvwChild, “internet-org”, “org”, 1, 2

Номера присваиваются по порядку, но можно и задавать ключ. Намного длиннее, но если вы хотите сделать рисунки для своих папок, а их 100 штук, то неужели вы каждый раз будете прописывать путь к файлу?

Вот мы везде пишем tvwChild, а неужели нельзя по-другому? Может, я хочу вставить следом или до некоторого узла новый — нет проблем.

TreeView1.Nodes.Add “internet-com”, tvwNext, “internet-tv”, “tv”

Добавить после домена com домен tv, а если перед? — правильно:

TreeView1.Nodes.Add “internet-com”, tvwPrevious, “internet-su”, “su”

А хотите первым поставить новый узел:

TreeView1.Nodes.Add “internet-com”, tvwFirst, “internet-to”, “to”

А последним:

TreeView1.Nodes.Add “internet-com”, tvwLast, “internet-med”, “med”

Вот и закончились варианты…

Здесь я привел лишь описание TreeView, методы же работы с ним, такие как поиск, сохранение, загрузка, намного сложнее.

## Создание всплывающей формы или отчета

Прежде чем перейти к выполнению следующих процедур, создайте нужные форму или отчет.

1. Откройте форму или отчет в режиме конструктора
2. Дважды щелкните [область выделения формы (*Область выделения формы –.* квадрат на пересечении линеек в верхнем левом углу окна конструктора форм. Используется для выполнения операций на уровне формы, таких как выделение формы.](javascript:AppendPopup(this,'defFormSelector_2')) или [область выделения отчета. *Область выделения отчета*– квадрат на пересечении линеек в верхнем левом углу окна конструктора отчетов. Используется для выполнения операций на уровне отчета, таких как выделение отчета](javascript:AppendPopup(this,'defReportSelector_3')), чтобы открыть [окно свойств. *Окно свойств* – окно, предназначенное для просмотра и изменения свойств таблиц, запросов, полей, форм, отчетов, страниц доступа к данным и элементов управления.)](javascript:AppendPopup(this,'defPropertySheet_4')).
3. Выберите значение *Да* для свойства *Всплывающее окно* (PopUp).
4. Выполните одно из следующих действий.

[Для немодальной всплывающей формы и отчета](javascript:ToggleDiv('divExpCollAsst_1'))



* 1. Если необходимо изменить размеры формы или отчета, то выберите значение *Тонкая* для свойства *Тип границы* (BorderStyle).
  2. Создайте макрос или процедуру обработки события для открытия всплывающей формы или отчета.

**Примечание.**  Пример процедуры обработки события, которая отображает немодальную всплывающую форму, находится в форме «Поставщики» учебной базы данных «Борей». Откройте базу данных «Борей», расположенную в подкаталоге Samples каталога Office, а затем откройте форму «Поставщики» в режиме конструктора. Нажмите кнопку Построить , расположенную рядом с ячейкой свойства Нажатие кнопки кнопки «Список товаров».



[Для модальной выплывающей формы или отчета](javascript:ToggleDiv('divExpCollAsst_2'))



* 1. В ячейке свойства **Модальное окно (Modal)** выберите значение **Да**.
  2. В ячейке свойства **Тип границы (BorderStyle)** выберите значение **Окно диалога**.
  3. Создайте макрос или процедуру обработки события для открытия всплывающей формы или отчета.

1. Свяжите макрос или процедуру обработки события с формой или отчетом, указав имя макроса или процедуры обработки события в ячейке соответствующего свойства события. Например, введите имя макроса или процедуры обработки события в ячейку свойства кнопки **Нажатие кнопки (OnClick)**.

Чтобы разместить всплывающую форму или отчет на экране нужным образом, откройте их в режиме конструктора, разверните окно Microsoft Access, выберите местоположение окна и сохраните форму или отчет. При открытии формы или отчета они появятся в этом месте (предполагается, что свойство Выравнивание по центру (AutoCenter) установлено в значение Нет и положение не задано в отрывающей их процедуре обработке события).

**Примечание.**  Чтобы использовать форму и как обычную форму, и как диалоговое окно, можно воспользоваться режимом окна «Окно диалога» макрокоманды «ОткрытьФорму» (OpenForm) для временного открытия формы в виде диалогового окна, а не применять эту процедуру.

## Работа со смарт-тегами в Microsoft Access

С помощью смарт-тегов можно сэкономить время при выполнении тех действий в приложении Microsoft Office Access 2003, для которых обычно предназначены другие программы. Смарт-теги можно присоединять к файлу в таблице или запросе, к элементу управления в форме, отчете или на странице доступа к данным.

**Пример.** Одним из **действий**, которые можно выполнить с помощью смарт-тега имени пользователя, является добавление имени в папку Microsoft Outlook «Контакты»

### Использование смарт-тегов

Добавить смарт-тег к полю или элементу управления можно с помощью установки свойства *Смарт-теги*. После добавления смарт-тега к полю или элементу управления при активизации ячейки этого поля или элемента управления появляется кнопка ***Действия*** для смарт-тегов . Нажмите эту кнопку, чтобы посмотреть меню действий, которые доступны для смарт-тегов.

### Получение дополнительных смарт-тегов

В приложении Access представлена лишь малая часть смарт-тегов. Чтобы найти новые смарт-теги в Интернете, выполните следующие действия.

* Создайте новую форму в режиме конструктора.
* Добавьте на форму надпись и нажмите кнопку ***Свойства*** на панели инструментов, чтобы открыть окно свойств элемента управления.
* Нажмите кнопку ***Построить*** в поле ***Смарт-теги***, чтобы открыть диалоговое окно ***Смарт-теги***.
* Нажмите кнопку ***Дополнительно***.

Можно найти дополнительные смарт-теги, созданные в корпорации Майкрософт, независимыми разработчиками либо специалистами отдела информационных технологий (IT), разрабатывающими смарт-теги и действия для определенных продуктов или служб. Например, если вы работаете в отделе продаж, используйте смарт-тег «имя продукта», щелкнув который, можно проверить количеств продуктов на складе или цены.

### Дополнительные сведения о разработке смарт-тегов для Access

Посетите центр разработчиков Office на веб-узле Microsoft Developer Network (MSDN), чтобы получить дополнительные сведения о создании смарт-тегов для Access.

## Создание масок ввода

|  |  |
| --- | --- |
| **Знаки маски ввода** | **Указываемый тип данных** |
| 0 | цифра (обязательный) |
| 9 | цифра (необязательный) |
| A | буква или цифра (обязательный) |
| a | буква или цифра (необязательный) |
| L | буква (обязательный) |
| ? | буква (необязательный) |
| # | цифра или пробел (необязательный — пробел, если оставляется пустым) |
| & | любой знак или пробел (обязательный) |

|  |  |
| --- | --- |
| **Пример маски ввода** | **Пример данных** |
| (000) 000-0000 | (206) 555-0248 |
| (999) 999-9999 | () 555-0248 |
| (000) AAA-AAAA | (206) 555-TELE |
| L????L?000L0 | GREENGR339М3 |
| L0L 0L0 | Т2Ф 8М4 |
| 00000-9999 | 98115-3007 |
| ISBN 0-&&&&&&&&&-0 | ISBN 1-55615-507-7 |

Создателя формы всегда должна беспокоить проблема, связанная с тем, что номера телефонов, даты и другие цифровые данные могут вводиться несколькими различными способами в одну и ту же форму. Например, один пользователь может ввести некоторую дату как 3/11/2007, а другой — как 03/11/07 или даже как 3 нояб. 2007. Подобная смесь форматов серьезно затрудняет использование данных.

Маски ввода предоставляют пользователям образцы, помогающие единообразным образом вводить числа, дефисы, косые линии и другие знаки в поля и поля со списками форм.

Например, маска ввода для даты может отображаться в элементе управления формы как ##/##/####, благодаря чему пользователи будут знать, в каком формате следует вводить соответствующие сведения (03/11/2007).

Чтобы создать маску ввода для элемента управления формы, следует выбрать этот элемент управления, щелкнуть кнопку **Свойства** и перейти на вкладку **Данные**. Далее необходимо щелкнуть ячейку свойств **Маска ввода**, а затем щелкнуть расположенные в ней три точки. Откроется мастер создания масок ввода.



В мастере предоставляется список общих типов данных, включая номер телефона, дату и время. При этом в базовые образцы масок ввода можно вносить изменения, а также просматривать результат их применения, а затем применять полученную новую маску к элементам управления формы.

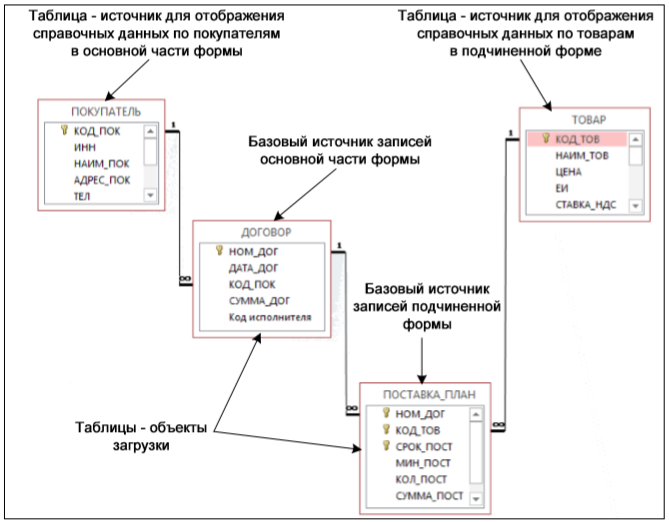
**Примечание.**    Мастер предложит также сохранить знаки форматирования, которые использовались при записи маски ввода (такие, как косые линии в приводившемся выше формате даты), но не являются данными. Пользователь может оставить эти знаки в сохраняемых данных или же указать, что они будут просто временно отображаться в соответствующем элементе управления. При этом необходимо *точно* знать, что выбранный вариант совместим с любыми вводившимися ранее и уже сохраненными данными.

## Разработка интерфейса для ввода, просмотра и корректировки документов

Прежде чем вводить, отображать или корректировать данные таблиц через экранную форму, надо ее спроектировать и сконструировать. Далее рассматриваются основы проектирования форм для построения удобного интерфейса пользователя для работы с документами.

### Этапы разработки интерфейса

1. Определение последовательности загрузки таблиц с документов
2. Проектирование интерфейса для ввода и корректировки документа
   1. Определение подсхемы данных
   2. Разработка макета
3. Создание интерфейса для ввода и корректировки документа
4. Доработка интерфейса
   1. Создание кнопок
   2. Ограничение доступа к данным таблицы
   3. Создание полей со списком
   4. Вычисления в документе
5. Работа с документами
6. Выборка документа по его идентификатору
7. Выборка документа по диапазону дат
8. Выборка документов с помощью фильтрации



Подсхема данных для формы ввода/вывода договоров фирмы

### Разработка макета

Изображение выглядит как стол

Автоматически созданное описание

Проект макета формы ДОГОВОРЫ С ПОКУПАТЕЛЯМИ

Вводить нужно только идентификаторы и количественные показатели. Справочные данные (наименования, цена и т. п.) не потребуется вводить из документа, они отобразятся в форме из ранее загруженных таблиц справочной информации. Отображение справочных данных позволяет осуществлять визуальный контроль правильности вводимых из документа данных.

ССЫЛКИ

* Создание формы в Access <https://support.office.com/ru-ru/article/Создание-формы-в-access-5d550a3d-92e1-4f38-9772-7e7e21e80c6b>

[Изображение выглядит как текст

Автоматически созданное описание](#_ПРИЛОЖЕНИЕ)