Использование помощника в Microsoft Access

Для использования в Microsoft Access объектов Microsoft Office, включая помощника, необходимо сначала установить ссылку на объектную библиотеку Microsoft Office. Находясь в режиме конструктора модуля, выберите в меню **Сервис** команду **Ссылки** и установите флажок **Microsoft Office 8.0 Object Library**.

Если эта ссылка не установлена, при попытке использования объектов из объектной библиотеки Microsoft Office будет выдано сообщение об ошибке компиляции «Не описан определяемый пользователем тип».

Примечание. В автономно выполняемой версии Microsoft Access помощник недоступен.

Общие сведения о помощнике (Microsoft Access)

Для использования в Microsoft Access объектов Microsoft Office необходимо сначала установить ссылку на объектную библиотеку Microsoft Office. Находясь в режиме конструктора модуля, выберите в меню **Сервис** команду **Ссылки** и установите флажок **Microsoft Office 8.0 Object Library**.

Если эта ссылка не установлена, при попытке использования объектов из объектной библиотеки Microsoft Office будет выдано сообщение об ошибке компиляции «Не описан определяемый пользователем тип».

После установки ссылки на объектную библиотеку Microsoft Office объекты, свойства и методы Microsoft Office отображаются в окне просмотра объектов. Для программирования помощника используются объекты **Assistant**, **Balloon**, **BalloonCheckbox** и **BalloonLabel**, а также семейства **BalloonCheckboxes** и **BalloonLabels**.

Следующие свойства объекта **Assistant** недоступны в Microsoft Access, хотя они могут быть доступны в других приложениях Microsoft Office:

- свойство **FeatureTips**
- свойство **HighPriorityTips**
- свойство **KeyboardShortcutTips**
- свойство **MouseTips**

Примечание. В автономно выполняемой версии Microsoft Access помощник недоступен.

Свойство FeatureTips (Microsoft Access)

Это свойство недоступно в Microsoft Access.

Свойство HighPriorityTips (Microsoft Access)

Это свойство недоступно в Microsoft Access.

Свойство KeyboardShortcutTips (Microsoft Access)

Это свойство недоступно в Microsoft Access.

Свойство MouseTips (Microsoft Access)

Это свойство недоступно в Microsoft Access.

Свойство Callback (Microsoft Access)

Свойство **Callback** используется при работе с немодальным окном подсказки. Немодальное окно подсказки позволяет пользователю перейти к приложению, в котором он работает, не закрывая это окно. Такое окно полезно для создания подсказки, которая должна быть постоянно видима при выполнении определенной последовательности действий в программе. Чтобы сделать окно подсказки немодальным, следует свойству **Mode** объекта **Balloon** присвоить значение **msoModeModeless**.

Свойство **Callback** задает функцию или макрос, выполняющиеся при нажатии пользователем кнопки в окне подсказки. Если в свойстве **Callback** задан вызов функции, эта функция может определить нажатую клавишу.

Свойству **Callback** присваивается имя функции или макроса. Например, в следующей строке задается значение свойства **Callback**:

```
Assistant.NewBalloon.Callback = "АнализФлажков"
```

Следует отметить, что значением свойства **Callback** может быть имя только процедуры **Function**; использование имени процедуры **Sub** не допускается. Процедура **Function** должна быть объявлена общей.

Помощник (Microsoft Access), пример

В следующей функции создается специальное окно подсказки помощника и определяется ответ пользователя:

```
Function AssistantBalloon(Optional varCheck As Variant, Optional varLabel
As Variant)
   Dim bch As BalloonCheckbox
   Dim intI As Integer
   Dim intReturn As Integer
   Dim strCheck(5) As String
   Dim strList As String

   ' Создает новое окно подсказки.
   Set bal = Assistant.NewBalloon
   ' Задает тип окна подсказки.
   bal.BalloonType = msoBalloonTypeButtons
   ' Указывает, что окно подсказки модальное.
   bal.Mode = msoModeModal

   ' Делает помощника видимым.
   If Assistant.Visible = False Then Assistant.Visible = True
   ' Проверяет, был ли передан в функцию первый аргумент.
   If Not IsMissing(varCheck) Then
      ' Если значение слишком велико, задает максимальное число флажков
(5).
      If varCheck > 6 Then
         varCheck = 5
      End If
      ' Присваивает свойству «Текст» (Text) символы алфавита.
      For intI = 1 To varCheck
         bal.Checkboxes(intI).Text = Chr(64 + intI)
      Next intI
   End If

   If Not IsMissing(varLabel) Then
      ' Если значение слишком велико, задает максимальное число меток (5).
      If varLabel > 6 Then
         varLabel = 5
      End If
      For intI = 1 To varLabel
         ' Присваивает свойству «Текст» (Text) символы алфавита.
         bal.Labels(intI).Text = Chr(64 + intI)
      Next intI
   End If
   ' Сохраняет возвращаемое значение.
   intReturn = bal.Show

   intI = 0
   ' Определяет установленные флажки.
   For Each bch In bal.Checkboxes
      If bch.Checked = True Then
         strCheck(intI) = bch.Text
         strList = strList & "'" & strCheck(intI) & "'" & Chr(32)
      End If
      intI = intI + 1
   Next
```

```
   If Len(strList) <> 0 Then
      MsgBox "Установлены флажки " & strList & "."
   End If
   ' Определить выделенные метки.
   If intReturn > 0 Then
      MsgBox "Выделены метки " & bal.Labels(intReturn).Text & "."
   End If
End Function
```

Эта функция может быть вызвана из окна отладки следующим образом:

```
? AssistantBalloon(4, 5)
```

Метод AddNew (Microsoft Access)

При использовании закладки в модуле Microsoft Access необходимо включить инструкцию **Option Compare Binary** в раздел описаний модуля. Закладка представляет из себя массив типа **Variant** данных**,** принадлежащих к типу **Byte,** поэтому для модуля необходимо выбрать двоичный способ сравнения строк. Если для проверки закладки используется способ сравнения текстовых строк, задающийся инструкцией **Option Compare Text**, или стандартная настройка **Option Compare Database**, то указатель текущей записи может быть установлен на неверную запись.

Метод Clone (Microsoft Access)

При использовании свойства **Bookmark** объекта **Recordset** в модуле Microsoft Access необходимо включить инструкцию **Option Compare Binary** в раздел описаний модуля. Закладка представляет из себя массив типа **Variant** данных, принадлежащих к типу **Byte,** поэтому для модуля необходимо выбрать двоичный способ сравнения строк. Если для проверки закладки используется способ сравнения текстовых строк, задающийся инструкцией **Option Compare Text**, или стандартная настройка **Option Compare Database**, то указатель текущей записи может быть установлен на неверную запись.

Метод Close (Microsoft Access)

Если в процедуре Visual Basic содержится объектная переменная, представляющая базу данных, которая является текущей базой данных, открытой в Microsoft Access, то вызов метода **Close** для этого объекта приведет к выходу переменной из области определения. Вызов метода **Close** не изменит состояния базы данных, открытой в окне базы данных Microsoft Access.

Метод CompactDatabase (Microsoft Access)

Для сжатия базы данных из окна Microsoft Access следует в меню **Сервис** выбрать команду **Служебные программы** и подкоманду **Сжать базу данных**. Для сжатия базы данных из программы Visual Basic следует воспользоваться методом DAO **CompactDatabase**.

Примечание. Выполнение метода **CompactDatabase** не приводит к полному преобразованию базы данных Microsoft Access от одной версии к другой. Преобразуется только формат данных. Объекты, определяемые в Microsoft Access, такие как формы и отчеты, не преобразуются. Для преобразования базы данных Microsoft Access от ранней версии к текущей следует выбрать в меню **Сервис** команду **Служебные программы** и подкоманду **Преобразовать базу данных**.

Метод CreateDatabase (Microsoft Access)

Метод **CreateDatabase** позволяет создать новую базу данных в программе Visual Basic. Новая база данных будет немедленно открыта, добавлена в семейство **Databases** и сохранена на диске.

Команда **CreateDatabase** не влияет на базу данных, открытую в окне базы данных Microsoft Access. Она остается открытой и является первой базой данных в семействе **Databases**. Функция **CurrentDb** возвращает ссылку на открытую в окне базу данных.

Для того чтобы открыть базу данных, созданную с помощью метода **CreateDatabase**, в окне базы данных, необходимо выбрать в меню **Файл** команду **Открыть**.

Метод CreateGroup (Microsoft Access)

После создания нового объекта **Group** и добавления его в семейство **Groups** объекта **Workspace** или **User** допускается проверка существования новой группы с помощью раскрывающегося списка **Имя** на вкладке **Группы** в диалоговом окне **Пользователи и группы**. Для того чтобы открыть это диалоговое окно, выберите в меню **Сервис** команду **Защита** и команду **Пользователи и группы** в подменю.

Метод CreateProperty (Microsoft Access)

Microsoft Access определяет ряд свойств объектов доступа к данным. Эти свойства не распознаются автоматически ядром базы данных Microsoft Jet. Для того чтобы задавать или возвращать в программах Visual Basic значение свойства, определяемого в Microsoft Access, необходимо явно добавить это свойство в семейство **Properties** объекта, к которому это свойство относится. Для этого следует сначала создать свойство с помощью метода **CreateProperty**, а затем добавить свойство в семейство **Properties**.

Например, в Microsoft Access описано свойство **Description** объекта **TableDef**. Если значение этого свойства уже не было задано в режиме конструктора таблиц, следует с помощью метода **CreateProperty** создать свойство, а затем добавить его в семейство **Properties**.

Свойство, определяемое в Microsoft Access, автоматически добавляется в семейство **Properties** при первом задании значения этого свойства в окне Microsoft Access. Если свойство уже определено таким способом, то нет необходимости явно добавлять его в семейство **Properties**.

При разработке программы, задающей значение свойства, определяемого в Microsoft Access, необходимо включить в нее блок обработки ошибок, который создает объект **Property**, представляющий данное свойство, и добавляет его в семейство **Properties**, если это свойство еще не включено в семейство.

При ссылках в программах Visual Basic на свойство, определяемое в Microsoft Access, необходимо явно сослаться на семейство **Properties**. Например, ссылка на свойство **AppTitle** после включения этого свойства в семейство **Properties** объекта **Database**, представляющего текущую базу данных, будет выглядеть следующим образом

```
Dim dbs As Database
Set dbs = CurrentDb
dbs.Properties!AppTitle = "Борей"
```

Примечание. Создавать и добавлять в семейство требуется только те свойства Microsoft Access, которые относятся к объектам доступа к данным. Значения других свойств Microsoft Access задаются в программе Visual Basic с помощью стандартного синтаксиса *объект.ИмяСвойства.*

Метод CreateUser (Microsoft Access)

После создания нового объекта **User** и добавления его в семейство **Users** объекта **Workspace** или **User**, допускается проверка существования нового имени пользователя с помощью раскрывающегося списка **Имя** на вкладке **Пользователи** в диалоговом окне **Пользователи и группы**. Для того чтобы открыть это диалоговое окно, выберите в меню **Сервис** команду **Защита** и команду **Пользователи и группы** в подменю.

Метод FillCache (Microsoft Access)

При использовании закладки в модуле Microsoft Access необходимо включить инструкцию **Option Compare Binary** в раздел описаний модуля. Закладка представляет из себя массив типа **Variant** данных, принадлежащих к типу **Byte,** поэтому для модуля необходимо выбрать двоичный способ сравнения строк. Если для проверки закладки используется способ сравнения текстовых строк, то указатель текущей записи может быть установлен на неверную запись. Инструкция **Option Compare Text** задает способ сравнения текстовых строк, аналогично устанавливаемой по умолчанию настройке **Option Compare Database**.

Методы FindFirst, FindLast, FindNext, FindPrevious (Microsoft Access)

При указании условий в методах группы **Find** необходимо проявлять аккуратность в ссылочных полях и элементах управления, а также правильно задавать строку поиска. Более подробное описание задания условий в функциях по подмножеству и использования кавычек в строковых выражениях содержится в разделе справки «Статистические функции по подмножеству».

При использовании закладки в модуле Microsoft Access необходимо включить инструкцию **Option Compare Binary** в раздел описаний модуля. Закладка представляет из себя массив типа **Variant** данных, принадлежащих к типу **Byte,** поэтому для модуля необходимо выбрать двоичный способ сравнения строк. Если для проверки закладки используется способ сравнения текстовых строк, задающийся инструкцией **Option Compare Text**, или стандартная настройка **Option Compare Database**, то указатель текущей записи может быть установлен на неверную запись.

Метод GetChunk (Microsoft Access)

После выполнения метода **GetChunk** и записи результата в строковую переменную функция Visual Basic **Len** позволяет определить число символов в строке. Определить размер строки в байтах позволяет функция **LenB**.

Метод Move (Microsoft Access)

При использовании закладки в модуле Microsoft Access необходимо включить инструкцию **Option Compare Binary** в раздел описаний модуля. Закладка представляет из себя массив типа **Variant** данных, принадлежащих к типу **Byte,** поэтому для модуля необходимо выбрать двоичный способ сравнения строк. Если для проверки закладки используется способ сравнения текстовых строк, задающийся инструкцией **Option Compare Text**, или стандартная настройка **Option Compare Database**, то указатель текущей записи может быть установлен на неверную запись.

Метод OpenDatabase (Microsoft Access)

В Microsoft Access для связывания переменной типа **Database** с текущей базой данных следует использовать функцию **CurrentDb,** как показано в следующем примере.

```
Dim dbsCurrent As Database
Set dbsCurrent = CurrentDb
```

Метод **OpenDatabase** следует использовать для открытия базы данных, отличной от текущей.

Метод Update (Microsoft Access)

При использовании закладки в модуле Microsoft Access необходимо включить инструкцию **Option Compare Binary** в раздел описаний модуля. Закладка представляет из себя массив типа **Variant** данных, принадлежащих к типу **Byte,** поэтому для модуля необходимо выбрать двоичный способ сравнения строк. Если для проверки закладки используется способ сравнения текстовых строк, задающийся инструкцией **Option Compare Text**, или стандартная настройка **Option Compare Database**, то указатель текущей записи может быть установлен на неверную запись.

МетодAddNew (Microsoft Access), пример

В следующем примере в таблице «Сотрудники» создается новая запись и сохраняются внесенные в нее изменения:

```
Sub AddNewRecord(ByVal rst As Recordset, _
   strLast As String, strFirst As String)

   With rst
      .AddNew                 ' Добавляет новую запись.
      !Фамилия = strLast      ' Заполнение данными.
      !Имя = strFirst
      .Update                 ' Сохранение изменений.
   End With
End Sub
```

Метод Append (Microsoft Access), пример

В следующем примере описывается и добавляется в семейство **Fields** объекта **TableDef** новый объект типа **Field**:

```
Sub NewField()
   Dim dbs As Database, tdf As TableDef, fld As Field

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   Set tdf = dbs.TableDefs!Сотрудники
   ' Создает новое поле в таблице «Сотрудники».
   Set fld = tdf.CreateField("Образование", dbText, 11)
   ' Добавляет поле и обновляет семейство.
   tdf.Fields.Append fld
   tdf.Fields.Refresh
   Set dbs = Nothing
End Sub
```

Методы AppendChunk и GetChunk (Microsoft Access), пример

В следующем примере в поле «Примечания» каждой записи таблицы «Сотрудники» добавляются некоторые сведения. Поле «Примечания» имеет тип MEMO. Для получения содержимого поля используется метод **GetChunk**, для добавления сведений и внесения измененного блока данных в поле «Примечания» используется метод **AppendChunk**.

```
Sub AddToMemo()
   Dim dbs As Database, rst As Recordset
   Dim fldNotes As Field, fldFirstName As Field
   Dim fldLastName As Field
   Dim lngSize As Long, strChunk As String

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Создает табличный объект Recordset.
   Set rst = dbs.OpenRecordset("Сотрудники")
   ' Возвращает ссылки на объекты Field.
   Set fldNotes = rst!Примечания
   Set fldFirstName = rst!Имя
   Set fldLastName = rst!Фамилия

   ' Цикл по всем записям в наборе записей.
   Do Until rst.EOF
      ' Проверяет существование данных в поле «Примечания».
      If IsNull(fldNotes.Value) Then
         ' Если данных нет, используется только метод AppendChunk.
         strChunk = fldFirstName _
            & " " & fldLastName & " отличный сотрудник."
         With rst
            .Edit
            !Примечания = strChunk
            .Update
            .MoveNext
         End With
      Else
         lngSize = Len(fldNotes)
         ' Получает существующие данные с помощью метода GetChunk.
         strChunk = fldNotes.GetChunk(0, lngSize)
         ' Изменяет данные.
         strChunk = strChunk & "  " & fldFirstName _
            & " " & fldLastName & " отличный работник."
         With rst
            .Edit                          ' Разрешает редактирование.
            !Примечания = ""               ' Инициализирует поле.
            !Примечания.AppendChunk strChunk    ' Добавляет измененные
данные.
            .Update                        ' Сохраняет изменения.
            .MoveNext                      ' Переходит к следующей записи.
         End With
      End If
   Loop
   rst.Close
   Set dbs = Nothing
End Sub
```

Методы BeginTrans, CommitTrans и Rollback (Microsoft Access), пример

В следующем примере в таблице «Сотрудники» изменяются названия должностей всех сотрудников, являющихся представителями. Метод **BeginTrans** начинает транзакцию, изолирующую все изменения в таблице «Сотрудники», а метод **CommitTrans** сохраняет внесенные изменения. Метод **Rollback** служит для отмены изменений, сохраненных с помощью метода **Update**. В примере также показано, как установить флаг для корректной обработки ошибок, возникающих во время транзакции.

Пока пользователь решает, сохранять ли ему изменения или нет, одна или несколько страниц таблицы остаются заблокированными. Поэтому, нижеследующую процедуру не рекомендуется применять на практике, она приведена только в качестве примера.

```
Sub ChangeTitle()
   Dim wsp As Workspace, dbs As Database, rst As Recordset
   Dim strName As String, strMessage As String, strPrompt As String
   Dim fInTrans As Boolean

   On Error GoTo ChangeTitleErr

   fInTrans = False
   strPrompt = "Изменить должность на 'Менеджер по продажам'?"
   ' Возвращает ссылку на заданный по умолчанию объект Workspace.
   Set wsp = DBEngine.Workspaces(0)
   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Создает табличный объект Recordset.
   Set rst = dbs.OpenRecordset("Сотрудники", dbOpenTable)
   ' Начало транзакции.
   wsp.BeginTrans
   fInTrans = True
   rst.MoveFirst
   Do Until rst.EOF
      If rst!Должность = "Представитель" Then
         strName = rst!Фамилия & ", " & rst!Имя
         strMessage = "Сотрудник: " & strName & vbCrLf & vbCrLf
         If MsgBox(strMessage & strPrompt, vbQuestion + vbYesNo, _
               "Изменить должность") = vbYes Then
            ' Разрешает редактирование.
            rst.Edit
            rst!Должность = "Менеджер по продажам"
            ' Сохраняет изменения.
            rst.Update
         End If
      End If
      ' Переходит к следующей записи.
      rst.MoveNext
   Loop
   If MsgBox("Сохранить все изменения?", vbQuestion + vbYesNo, _
         " Сохранение изменений") = vbYes Then
      wsp.CommitTrans         ' Принимает изменения.
   Else
      wsp.Rollback            ' Отменяет изменения.
   End If
ChangeTitleExit:
   rst.Close
   Set dbs = Nothing
```

```
   Set wsp = Nothing
   Exit Sub
ChangeTitleErr:
   MsgBox "Ошибка!"
   If fInTrans Then
      wsp.Rollback
   End If
   Resume ChangeTitleExit
End Sub
```

Методы CancelUpdate и Update (Microsoft Access), пример

В следующем примере в таблицу «Сотрудники» добавляется новая запись, вводятся значения, и пользователю предлагается сохранить изменения. Если пользователь отказывается от сохранения, метод **Update** не вызывается.

```
Sub NewRecord()
   Dim dbs As Database, rst As Recordset

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   Set rst = dbs.OpenRecordset("Сотрудники")
   With rst
      ' Добавляет новую запись в конец объекта Recordset.
      .AddNew
      !Фамилия = "Иванов"    ' Ввести данные.
      !Имя = "Александр"
   End With
   ' Предлагает пользователю сохранить изменения.
   If MsgBox("Сохранить изменения?", vbYesNo) = vbNo Then
      ' Если пользователь отказался, отменяет изменения.
      rst.CancelUpdate
   Else
      ' Если пользователь согласился, сохраняет изменения.
      rst.Update
   End If
   rst.Close
   Set dbs = Nothing
End Sub
```

Метод Clone (Microsoft Access), пример

В следующем примере из таблицы «Сотрудники» с помощью инструкции SQL создается объект **Recordset**, а затем используется метод **Clone** для получения копии объекта **Recordset** с тем, чтобы можно было использовать закладки совместно в двух объектах. Этот способ особенно важен при сравнении результатов запроса одновременно из нескольких точек.

Эта функция читает и сохраняет в строковой переменной значение свойства **Bookmark** текущей записи – в данном случае вторую запись исходного объекта **Recordset**. Свойству **Bookmark** копии объекта **Recordset** присваивается эта строка, так же помечая вторую запись в качестве текущей. Текущие значения поля «Фамилия» двух объектов совпадают, что можно проверить в окне отладки.

```
Sub CreateClone()
   Dim dbs As Database
   Dim rstEmployees As Recordset, rstDuplicate As Recordset
   Dim fldName As Field, varBook As Variant

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Открывает динамический объект Recordset.
   Set rstEmployees = dbs.OpenRecordset("SELECT * FROM Сотрудники " _
      & " ORDER BY Фамилия")
   ' Создает копию объекта Recordset.
   Set rstDuplicate = rstEmployees.Clone
   Set fldName = rstEmployees.Fields!Фамилия
   ' Устанавливает текущую запись.
   rstEmployees.MoveFirst
   ' Переходит ко второй записи.
   rstEmployees.MoveNext
   ' Получает значение свойства Bookmark и выводит значение текущего поля.
   If rstEmployees.Bookmarkable Then
      varBook = rstEmployees.Bookmark
      Debug.Print fldName.value
   Else
      ' Если объект Recordset не поддерживает закладки, завершает
процедуру.
      ExitCreateClone
   End If
   ' Присваивает свойству копии Bookmark полученное значение.
   rstDuplicate.Bookmark = varBook
   Debug.Print fldName.value

ExitCreateClone:
   rstEmployees.Close
   rstDuplicate.Close
   Set dbs = Nothing
End Sub
```

Метод Close (Microsoft Access), пример

В следующем примере создается объект **Database**, указывающий на текущую базу данных, а также на основе таблицы «Клиенты» открывается табличный объект **Recordset**. В этой процедуре метод **Close** применяется для освобождения занимаемой памяти к объектной переменной типа **Recordset**. Затем инструкция **Set** с ключевым словом **Nothing** освобождает ресурсы, занимаемые объектной переменной типа **Database**.

Метод **Close** объекта **Database** может также применяться для закрытия базы данных и освобождения памяти. Метод **Close** объекта **Database** в действительности не закрывает открытую в Microsoft Access базу данных, он только освобождает ресурсы, используемые объектной переменной типа **Database**.

Эквивалентным способом освобождения памяти является вызов метода **Close** объекта и присваивание объектной переменной значения **Nothing**.

```
Sub UseClose()
   Dim dbs As Database, rst As Recordset

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Создает табличный объект Recordset.
   Set rst = dbs.OpenRecordset("Клиенты")
   .
   .
   .
   ' Закрывает объект Recordset, чтобы освободить память.
   rst.Close
   ' Освобождает память, занимаемую объектной переменной.
   Set dbs = Nothing
End Sub
```

Метод CopyQueryDef (Microsoft Access), пример

В следующем примере метод **CopyQueryDef** используется для получения копии объекта **QueryDef**, представляющего запрос «Счета», и вывода свойства **SQL** этого объекта **QueryDef**.

```
Sub GetQueryDefCopy()
   Dim dbs As Database, rst As Recordset
   Dim qdfOriginal As QueryDef, qdfCopy As QueryDef

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Возвращает ссылку на запрос «Счета».
   Set qdfOriginal = dbs.QueryDefs!Счета
   ' Открывает динамический объект Recordset.
   Set rst = qdfOriginal.OpenRecordset
   ' Получает копию исходного объекта QueryDef.
   Set qdfCopy = rst.CopyQueryDef
   ' Выводит значение свойства SQL для копии объекта.
   Debug.Print qdfCopy.SQL
   rst.Close
   Set dbs = Nothing
End Sub
```

Метод CreateDatabase (Microsoft Access), пример

В следующем примере метод **CreateDatabase** используется для создания новой зашифрованной базы данных с именем НоваяБД.mdb:

```
Sub NewDatabase()
   Dim wspDefault As Workspace, dbs As Database
   Dim tdf As TableDef, fld1 As Field, fld2 As Field
   Dim idx As Index, fldIndex As Field

   Set wspDefault = DBEngine.Workspaces(0)
   ' Создает новую зашифрованную базу данных.
   Set dbs = wspDefault.CreateDatabase("НоваяБД.mdb", _
      dbLangGeneral, dbEncrypt)
   ' Создает новую таблицу с двумя полями.
   Set tdf = dbs.CreateTableDef("Контакты")
   Set fld1 = tdf.CreateField("КодКонтакта", dbLong)
   fld1.Attributes = fld1.Attributes + dbAutoIncrField
   Set fld2 = tdf.CreateField("НазваниеКонтакта", dbText, 50)
   ' Добавляет поля.
   tdf.Fields.Append fld1
   tdf.Fields.Append fld2
   ' Создает индекс по ключевому полю.
   Set idx = tdf.CreateIndex("КлючевоеПоле")
   Set fldIndex = idx.CreateField("КодКонтакта", dbLong)
   ' Добавляет поля индекса.
   idx.Fields.Append fldIndex
   ' Задает значение свойства Primary.
   idx.Primary = True
   ' Добавляет индекс.
   tdf.Indexes.Append idx
   ' Добавляет объект TableDef.
   dbs.TableDefs.Append tdf
   dbs.TableDefs.Refresh
   Set dbs = Nothing
End Sub
```

Метод CreateField (Microsoft Access), пример

В следующем примере создается новая таблица с двумя новыми полями. Одно из этих полей имеет тип «Счетчик». Это поле становится ключевым полем таблицы.

```
Sub NewTable()
   Dim dbs As Database
   Dim tdf As TableDef, fld1 As Field, fld2 As Field
   Dim idx As Index, fldIndex As Field

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Создает новую таблицу с двумя новыми полями.
   Set tdf = dbs.CreateTableDef("Контакты")
   Set fld1 = tdf.CreateField("КодКонтакта", dbLong)
   fld1.Attributes = fld1.Attributes + dbAutoIncrField
   Set fld2 = tdf.CreateField("НазваниеКонтакта", dbText, 50)
   ' Добавляет поля.
   tdf.Fields.Append fld1
   tdf.Fields.Append fld2
   ' Создает индекс по ключевому полю.
   Set idx = tdf.CreateIndex("Ключевое поле")
   Set fldIndex = idx.CreateField("КодКонтакта", dbLong)
   ' Добавляет поля индекса.
   idx.Fields.Append fldIndex
   ' Задает значение свойства Primary.
   idx.Primary = True
   ' Добавляет индекс.
   tdf.Indexes.Append idx
   ' Добавляет объект TableDef.
   dbs.TableDefs.Append tdf
   dbs.TableDefs.Refresh
   Set dbs = Nothing
End Sub
```

Метод CreateIndex (Microsoft Access), пример

В следующем примере в таблице «Сотрудники» создается новый объект типа **Index**. Новый индекс состоит из двух полей: «Фамилия» и «Имя».

```
Sub NewIndex()
   Dim dbs As Database, tdf As TableDef
   Dim idx As Index
   Dim fldLastName As Field, fldFirstName As Field

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Возвращает ссылку на таблицу «Сотрудники».
   Set tdf = dbs.TableDefs!Сотрудники
   ' Возвращает ссылку на новый индекс.
   Set idx = tdf.CreateIndex("ФИО")
   ' Создает и добавляет поля индекса.
   Set fldLastName = idx.CreateField("Фамилия", dbText)
   Set fldFirstName = idx.CreateField("Имя", dbText)
   idx.Fields.Append fldLastName
   idx.Fields.Append fldFirstName
   ' Добавляет объект Index и обновляет семейство.
   tdf.Indexes.Append idx
   tdf.Indexes.Refresh
   Set dbs = Nothing
End Sub
```

Метод CreateProperty (Microsoft Access), пример

В первой процедуре следующего примера функция SetAccessProperty является общей функцией для задания значения свойства. Если необходимое свойство уже имеет соответствующий объект **Property** в семействе **Properties**, эта функция просто присваивает свойству требуемое значение. Если же у свойства нет соответствующего объекта **Property** в семействе **Properties**, функция создает такой объект и добавляет его в семейство.

Во второй процедуре с помощью функции SetAccessProperty задается значение свойства **Subject**. Свойство **Subject** является свойством базы данных и может быть также изменено на вкладке **Документ** диалогового окна **Свойства**, вызываемого по команде **Свойства базы данных** из меню **Файл**.

Свойство **Subject** относится к объекту DAO – объекту **Document** с именем SummaryInfo, определенному в Microsoft Access. Если для объекта **Document** SummaryInfo свойство **Subject** еще не существует в семействе **Properties**, функция SetAccessProperty добавляет его туда.

```
Function SetAccessProperty(obj As Object, strName As String, _
      intType As Integer, varSetting As Variant) As Boolean
   Dim prp As Property
   Const conPropNotFound As Integer = 3270

   On Error GoTo ErrorSetAccessProperty
   ' Явная ссылка на семейство Properties.
   obj.Properties(strName) = varSetting
   obj.Properties.Refresh
   SetAccessProperty = True

ExitSetAccessProperty:
   Exit Function

ErrorSetAccessProperty:
   If Err = conPropNotFound Then
      ' Создает свойство, задает тип и начальное значение.
      Set prp = obj.CreateProperty(strName, intType, varSetting)
      ' Добавляет объект Property в семейство Properties.
      obj.Properties.Append prp
      obj.Properties.Refresh
      SetAccessProperty = True
      Resume ExitSetAccessProperty
   Else
      MsgBox Err & ": " & vbCrLf & Err.Description
      SetAccessProperty = False
      Resume ExitSetAccessProperty
   End If
End Function
```

В следующей процедуре для задания значения свойства **Subject** используется метод SetAccessProperty.

```
Sub CallPropertySet()
   Dim dbs As Database, ctr As Container, doc As Document
   Dim blnReturn As Boolean

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Возвращает ссылку на контейнер Databases.
```

```
   Set ctr = dbs.Containers!Databases
   ' Возвращает ссылку на документ SummaryInfo.
   Set doc = ctr.Documents!SummaryInfo
   blnReturn = SetAccessProperty(doc, _
      "Subject", dbText, "Деловые контакты")
   ' Вычисляет возвращаемое значение.
   If blnReturn = True Then
      Debug.Print "Значение свойства успешно задано."
   Else
      Debug.Print "Значение свойства задать не удалось."
   End If
End Sub
```

Метод CreateQueryDef (Microsoft Access), пример

В следующем примере создается новый объект **QueryDef**, а затем открывается запрос в режиме таблицы:

```
Sub NewQuery()
   Dim dbs As Database, qdf As QueryDef, strSQL As String

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   strSQL = "SELECT * FROM Сотрудники WHERE [ДатаНайма] >= #1-1-93#"
   ' Создает новый запрос.
   Set qdf = dbs.CreateQueryDef("НедавноНанятые", strSQL)
   DoCmd.OpenQuery qdf.Name
   Set dbs = Nothing
End Sub
```

Метод CreateRelation (Microsoft Access), пример

В следующем примере создается новый объект **Relation**, определяющий отношение между таблицами «Типы» и «Товары». Таблица «Типы» является главной в отношении, а таблица «Товары» – внешней таблицей с ключом. Поле «Тип» является ключевым полем таблицы «Типы» и внешним полем таблицы «Товары».

Для проверки этого примера в демонстрационной базе данных «Борей» следует выбрать команду **Схема данных** из меню **Сервис** и удалить отношение между таблицами «Типы» и «Товары». Затем закрыть окно схемы данных, сохранив текущую конфигурацию. После этого следует выполнить следующую процедуру и снова открыть окно схемы данных для просмотра нового отношения.

```
Sub NewRelation()
   Dim dbs As Database, rel As Relation, fld As Field

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Создает новый объект Relation и задает внешнюю таблицу.
   Set rel = dbs.CreateRelation("ТипыТовары", "Типы", "Товары")
   ' Устанавливает атрибуты для обеспечения целостности данных.
   rel.Attributes = dbRelationUpdateCascade And dbRelationDeleteCascade
   ' Создает поле в объекте Relation.
   Set fld = rel.CreateField("Тип")
   ' Задает имя поля внешней таблицы.
   fld.ForeignName = "Тип"
   ' Добавляет объект Field в семейство Fields объекта Relation.
   rel.Fields.Append fld
   ' Добавляет объект Relation в семейство Relations.
   dbs.Relations.Append rel
   dbs.Relations.Refresh
   Set dbs = Nothing
End Sub
```

Метод CreateTableDef (Microsoft Access), пример

В следующем примере создается и добавляется в семейство **TableDefs** текущей базы данных новый объект **TableDef**.

```
Sub NewTable()
   Dim dbs As Database, tdf As TableDef, fld As Field

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Возвращает объектную переменную TableDef, указывающую на новую
таблицу.
   Set tdf = dbs.CreateTableDef("Контакты")
   ' Определяет в таблице новое поле.
   Set fld = tdf.CreateField("НазваниеКонтакта", dbText, 40)
   ' Добавляет объект Field в семейство Fields объекта TableDef.
   tdf.Fields.Append fld
   tdf.Fields.Refresh
   ' Добавляет объект TableDef в семейство TableDefs базы данных.
   dbs.TableDefs.Append tdf
   dbs.TableDefs.Refresh
   Set dbs = Nothing
End Sub
```

Метод Delete (Microsoft Access), пример

В следующем примере в таблице создается а затем удаляется поле:

```
Sub DeleteField()
   Dim dbs As Database, tdf As TableDef
   Dim fldInitial As Field

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   Set tdf = dbs.TableDefs!Сотрудники
   ' Создает новый объект Field.
   Set fldInitial = tdf.CreateField("ВременноеПоле", dbText, 2)
   ' Добавляет новый объект Field.
   tdf.Fields.Append fldInitial
   ' Обновляет семейство Fields.
   tdf.Fields.Refresh
   ' Удаляет новый объект Field.
   tdf.Fields.Delete fldInitial.Name
   tdf.Fields.Refresh
   Set dbs = Nothing
End Sub
```

Метод Edit (Microsoft Access), пример

В следующем примере открывается объект **Recordset** и ищутся все записи, удовлетворяющие условию поиска для поля «Должность». Затем для подготовки записей к редактированию используется процедура **Edit**, изменяется должность и изменения сохраняются.

```
Sub ChangeTitle()
   Dim dbs As Database, rst As Recordset
   Dim strCriteria As String, strNewTitle As String

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Задает условие поиска.
   strCriteria = "Должность = 'Представитель'"
   strNewTitle = "Менеджер по продажам"
   ' Создает динамический объект Recordset.
   Set rst = dbs.OpenRecordset("Сотрудники", dbOpenDynaset)
   ' Находит первое вхождение.
   rst.FindFirst strCriteria
   ' Выполняет цикл, пока есть совпадения.
   Do Until rst.NoMatch
      With rst
         .Edit                      ' Разрешает изменения.
         !Должность = strNewTitle   ' Изменяет должность.
         .Update                    ' Сохраняет изменения.
         .FindNext strCriteria      ' Находит следующее вхождение.
      End With
   Loop
   rst.Close
   Set dbs = Nothing
End Sub
```

Совет. Для изменения данных эффективнее использовать запрос на обновление. Например, следующая программа выполняет ту же задачу:

```
Sub ChangeTitleSQL()
   Dim dbs As Database
   Dim strSQL As String

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Создает строку SQL.
   strSQL = "UPDATE Сотрудники SET Должность = 'Менеджер по продажам' " _
      & "WHERE Должность = 'Представитель' "
   ' Выполняет запрос на изменение.
   dbs.Execute strSQL
   ' Возвращает число обновленных записей.
   Debug.Print dbs.RecordsAffected
   Set dbs = Nothing
End Sub
```

Метод Execute (Microsoft Access), пример

В следующем примере выполняется запрос на изменение и отображается число измененных записей:

```
Sub RecordsUpdated()
   Dim dbs As Database, qdf As QueryDef
   Dim strSQL As String

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   strSQL = "UPDATE Сотрудники SET Должность = " _
      & "'Старший представитель' " _
      & "WHERE Должность = 'Представитель';"
   ' Создает новый запрос QueryDef.
   Set qdf = dbs.CreateQueryDef("ОбновлениеДолжности", strSQL)
   ' Выполняет запрос QueryDef.
   qdf.Execute
   Debug.Print qdf.RecordsAffected
   Set dbs = Nothing
End Sub
```

Свойство Размер поля (FieldSize) (Microsoft Access), пример

В следующем примере свойство **Размер поля (FieldSize)** используется для получения размера в байтах двух полей таблицы «Сотрудники». Поле «Примечания» имеет тип MEMO, а поле «Фотография» содержит двоичные данные (объект OLE).

```
Sub GetFieldSize()
   Dim dbs As Database, rst As Recordset
   Dim fldNotes As Field, fldPhoto As Field
   Dim strSQL As String

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Создает инструкцию SQL для получения полей «Примечания» и
«Фотография».
   strSQL = "SELECT Примечания, Фотография FROM Сотрудники;"
   ' Создает динамический объект Recordset.
   Set rst = dbs.OpenRecordset(strSQL)
   Set fldNotes = rst!Примечания
   Set fldPhoto = rst!Фотография
   ' Переходит к первой записи.
   rst.MoveFirst
   Debug.Print "Размер поля 'Примечания':"; "  "; " Размер поля
'Фотография':"
   ' Выводит размеры полей для каждой записи объекта Recordset.
   Do Until rst.EOF
      Debug.Print fldNotes.FieldSize; "        "; fldPhoto.FieldSize
      rst.MoveNext
   Loop
   rst.Close
   Set dbs = Nothing
End Sub
```

Методы FindFirst, FindLast, FindNext и FindPrevious (Microsoft Access, пример)

В следующем примере создается динамический объект **Recordset**, а затем с помощью метода **FindFirst** находится первая запись, удовлетворяющая указанному условию. После этого находятся все остальные записи, удовлетворяющие условию.

```
Sub FindRecord()
   Dim dbs As Database, rst As Recordset
   Dim strCriteria As String

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Задает условие поиска.
   strCriteria = "[СтранаПолучателя] = 'Великобритания' And
[ДатаРазмещения] >= #1-1-95#"
   ' На основе таблицы «Заказы» создает динамический объект Recordset.
   Set rst = dbs.OpenRecordset("Заказы", dbOpenDynaset)
   ' Находит первую удовлетворяющую условию поиска запись.
   rst.FindFirst strCriteria
   ' Проверяет, найдена ли запись.
   If rst.NoMatch Then
      MsgBox "Запись не найдена."
   Else
      ' Находит остальные удовлетворяющие условию поиска записи.
      Do Until rst.NoMatch
         Debug.Print rst!СтранаПолучателя; "   "; rst!ДатаРазмещения
         rst.FindNext strCriteria
      Loop
   End If
   rst.Close
   Set dbs = Nothing
End Sub
```

Метод GetRows (Microsoft Access), пример

В следующем примере метод **GetRows** возвращает двухмерный массив, содержащий все строки данных объекта **Recordset**:

```
Sub RowsArray()
   Dim dbs As Database, rst As Recordset, strSQL As String
   Dim varRecords As Variant, intI As Integer, intJ As Integer

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Создает инструкцию SQL, возвращающую указанные поля.
   strSQL = "SELECT Имя, Фамилия, ДатаНайма " _
      & "FROM Сотрудники"
   ' Открывает динамический объект Recordset.
   Set rst = dbs.OpenRecordset(strSQL)
   ' Переходит в конец набора записей.
   rst.MoveLast
   ' Возвращается к первой записи.
   rst.MoveFirst
   ' Помещает все строки в массив.
   varRecords = rst.GetRows(rst.RecordCount)
   ' Находит верхнюю границу второго измерения массива.
   For intI = 0 To UBound(varRecords, 2)
      Debug.Print
      ' Находит верхнюю границу первого измерения массива.
      For intJ = 0 To UBound(varRecords, 1)
         ' Отображает данные каждой строки массива.
         Debug.Print varRecords(intJ, intI)
      Next intJ
   Next intI
   rst.Close
   Set dbs = Nothing
End Sub
```

Метод Move (Microsoft Access), пример

В следующем примере с помощью метода **Move** выполняется перемещение на две строки вперед в объекте **Recordset**:

```
Sub MoveForward()
   Dim dbs As Database, rst As Recordset
   Dim strCriteria As String

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Открывает динамический объект Recordset.
   Set rst = dbs.OpenRecordset("SELECT * FROM Заказы " _
      & "ORDER BY СтранаПолучателя;")
   rst.MoveLast
   rst.MoveFirst
   ' Проверяет  число записей в объекте Recordset.
   If rst.RecordCount > 2 Then
      ' Перемещается на две строки вперед.
      rst.Move 2
      Debug.Print rst!СтранаПолучателя
   End If
   rst.Close
   Set dbs = Nothing
End Sub
```

Методы MoveFirst, MoveLast, MoveNext и MovePrevious (Microsoft Access), пример

В следующем примере для вычисления количества записей в объекте **Recordset** объект заполняется с помощью метода **MoveLast**. Затем с помощью метода **MoveFirst** указатель текущей записи перемещается на первую запись объекта **Recordset**. После этого пользователю предлагается ввести число записей, на которое будет сделан переход вперед.

```
Sub MoveThroughRecords()
   Dim dbs As Database, rst As Recordset, intI As Integer
   Dim strNumber As String

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   Set rst = dbs.OpenRecordset("Товары")
   ' Заполняет объект Recordset.
   rst.MoveLast
   ' Возвращается к первой записи.
   rst.MoveFirst
   ' Получает число, меньшее количества записей.
   strNumber = InputBox("Пожалуйста введите число, меньшее чем " _
      & rst.RecordCount & ".")
   ' Перемещается на указанное число записей вперед.
   For intI = 1 To strNumber
      rst.MoveNext
   Next intI
   Debug.Print rst!Марка
   rst.Close
   Set dbs = Nothing
End Sub
```

Метод OpenDatabase (Microsoft Access), пример

В следующем примере возвращается переменная типа **Database**, указывающая на текущую базу данных. Затем с помощью метода **OpenDatabase** открывается новая база данных с именем «Другая.mdb». После этого процедура перечисляет все объекты **TableDef** в обеих базах данных.

Для проверки этого примера следует создать базу данных с именем «Другая.mdb», закрыть ее, и поместить в папку, в которой находится база данных с данной программой.

```
Sub OpenAnother()
   Dim wsp As Workspace
   Dim dbs As Database, dbsAnother As Database
   Dim tdf As TableDef

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Возвращает ссылку на заданную по умолчанию рабочую область.
   Set wsp = DBEngine.Workspaces(0)
   ' Возвращает ссылку на файл Другая.mdb.
   Set dbsAnother = wsp.OpenDatabase("Другая.mdb")
   ' Перечисляет все объекты TableDef в каждой базе данных.
   Debug.Print dbs.Name & ":"
   For Each tdf in dbs.TableDefs
      Debug.Print tdf.Name
   Next tdf
   Debug.Print
   Debug.Print dbsAnother.Name & ":"
   For Each tdf in dbsAnother.TableDefs
      Debug.Print tdf.Name
   Next tdf
   Set dbs = Nothing
   Set dbsAnother = Nothing
End Sub
```

Метод OpenRecordset (Microsoft Access), пример

В следующем примере открывается динамический объект **Recordset** и отображается число записей в этом объекте.

```
Sub UKOrders()
   Dim dbs As Database, rst As Recordset
   Dim strSQL As String

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   strSQL = "SELECT * FROM Заказы WHERE [СтранаПолучателя] =
'Великобритания'"
   Set rst = dbs.OpenRecordset(strSQL)
   rst.MoveLast
   Debug.Print rst.RecordCount
   rst.Close
   Set dbs = Nothing
End Sub
```

Метод Refresh (Microsoft Access), пример

В следующем примере выполняется обновление семейства **Indexes** объекта **TableDef**. Метод **Refresh** используется в многопользовательской среде для отображения изменений, внесенных другими пользователями в объекты **Index** семейства **Indexes** объекта **TableDef**.

```
Sub RefreshIndex()
   Dim dbs As Database, tdf As TableDef
   Dim idx As Index, fld As Field

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   Set tdf = dbs.TableDefs!Сотрудники
   tdf.Indexes.Refresh
   For Each idx In tdf.Indexes
      Debug.Print idx.Name; ":"
      For Each fld In idx.Fields
         Debug.Print "   "; fld.Name
      Next fld
   Next idx
   Set dbs = Nothing
End Sub
```

Метод Seek (Microsoft Access), пример

В следующем примере в таблице «Сотрудники» создается новый объект типа **Index**. Новый индекс состоит из двух полей: «Фамилия» и «Имя». Затем для нахождения заданной записи используется метод **Seek**.

```
Sub NewIndex()
   Dim dbs As Database, tdf As TableDef, idx As Index
   Dim fldLastName As Field, fldFirstName As Field, rst As Recordset

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Возвращает ссылку на таблицу «Сотрудники».
   Set tdf = dbs.TableDefs!Сотрудники
   ' Возвращает объект типа Index, указывающий на новый индекс.
   Set idx = tdf.CreateIndex("ФИО")
   ' Создает и добавляет индексные поля.
   Set fldLastName = idx.CreateField("Фамилия", dbText)
   Set fldFirstName = idx.CreateField("Имя", dbText)
   idx.Fields.Append fldLastName
   idx.Fields.Append fldFirstName
   ' Добавляет объект Index.
   tdf.Indexes.Append idx
   tdf.Indexes.Refresh
   ' Открывает табличный объект Recordset.
   Set rst = dbs.OpenRecordset("Сотрудники")
   ' Делает новый индекс текущим.
   rst.Index = idx.Name
   ' Указывает искомую запись.
   rst.Seek "=", "Иванов", "Александр"
   If rst.NoMatch Then
      Debug.Print "Поиск неудачен!"
   Else
      Debug.Print "Поиск удачен."
   End If
   rst.Close
   Set dbs = Nothing
End Sub
```

Объект Error (Microsoft Access)

Значением свойства **Description** объекта **Error** является краткое описание ошибки объектов доступа к данным (DAO). Данное свойство используют только в случае возникновения ошибки DAO.

Метод **AccessError** позволяет получить описание конкретной ошибки DAO Microsoft Access по номеру без фактического возникновения ошибки. Например, для задания описания ошибки под номером 3021 используется следующий формат вызова метода **AccessError**:

```
Debug.Print AccessError(3021)
```

В этом случае Microsoft Access выведет в окно отладки строку описания, связанную с ошибкой DAO 3021, «Текущая запись отсутствует».

Объект Recordset (Microsoft Access)

При использовании в приложениях, созданных в версиях Microsoft Access 1.*x* или 2.0, для указания на объект **Recordset** оператора **.** (точка), необходимо его изменение на оператор **!** (восклицательный знак). Для использования старого синтаксиса **.** (точка) необходимо в диалоговом окне **Ссылки**, вызываемом командой **Ссылки** меню **Сервис**, в режиме конструктора сделать ссылку на библиотеку совместимости Microsoft DAO 2.5/3.5.

Объект Container, семейство Containers (Microsoft Access)

В дополнение к объектам **Container** (контейнер), определенным в ядре базы данных Microsoft Jet, в Microsoft Access определены еще четыре объекта **Container**.

Имя контейнера	Описываемые объекты
Forms	Сохраненные формы
Modules	Сохраненные модули
Reports	Сохраненные отчеты
Scripts	Сохраненные макросы

Объекты **Container**, определенные в Microsoft Access, представляют объекты базы данных Microsoft Access, а не объекты доступа к данным (DAO). Эти объекты **Container** передают в ядро базы данных Jet информацию об объектах Microsoft Access. Эта информация используется ядром Jet для обеспечения той же системы защиты объектов базы данных Microsoft Access, которая реализуется ядром для объектов доступа к данным.

И объекты **Container**, определенные в Microsoft Access, и объекты **Container**, определенные в ядре Jet, включаются в семейство **Containers** Microsoft Access.

Семейство **Documents** объекта **Container** содержит объекты **Document**, представляющие отдельные объекты, тип которых описывается в объекте **Container**. Например, объект **Container** с именем Forms содержит семейство **Documents**, в которое может входить объект **Document**, соответствующий форме **Заказы**.

Объекты **Container** могут быть использованы для установления и обеспечения разрешения на доступ к объектам базы данных Microsoft Access. Для того чтобы установить разрешение на доступ к объекту **Container**, следует указать в свойстве **UserName** объекта **Container** имя существующего объекта **User** (пользователь) или **Group** (группа). После этого следует определить для объекта **Container** свойство **Permissions**.

Примечание. Не следует путать отдельные типы объектов **Container** с одноименными типами семейств. Объекты **Container** с именами **Forms** и **Reports** содержат семейство **Documents**, в которое входят отдельные объекты **Document**, представляющие каждую сохраненную форму или отчет. В семейства **Forms** и **Reports** включаются только открытые формы и отчеты.

Объект Database, семейство Databases (Microsoft Access)

При обращении к объектам доступа к данным из Microsoft Access часто приходится определять объектную переменную типа **Database**, представляющую текущую базу данных. Функция **CurrentDb** возвращает объект **Database**, соответствующий текущей открытой базе данных. Этот объект автоматически добавляется в семейство **Databases**.

Предположим, например, что вы работаете в Microsoft Access с учебной базой данных «Борей». Для того чтобы создать объект **Database**, указывающий на эту базу данных, следует сначала определить объектную переменную типа **Database**, а затем присвоить ей объект **Database**, возвращаемый функцией **CurrentDb**.

```
Dim dbs As Database
Set dbs = CurrentDb
```

Для того чтобы использовать текущую базу данных, нет необходимости знать имя базы данных или ее положение в семействе **Databases**. Если потребуется узнать имя текущей базы данных, следует проверить значение свойства **Name** объекта **Database**, содержащее путь и имя файла базы данных. Для проверки положения базы данных в семействе **Databases** следует перебрать компоненты семейства.

В окне Microsoft Access может быть открыта только одна база данных. В программе Visual Basic, однако, пользователь имеет возможность создать несколько независимых объектных переменных типа **Database**, представляющих разные открытые базы данных. Это позволяет одновременно работать в программе с несколькими базами данных. Допускается также создание нескольких объектных переменных типа **Database**, указывающих на текущую базу данных.

Примечание. В программе Visual Basic рекомендуется использовать функцию **CurrentDb**, возвращающую объект **Database**, представляющий текущую базу данных, вместо синтаксиса `DBEngine(0)(0)`. Функция **CurrentDb** создает новый экземпляр текущей базы данных, тогда как синтаксис `DBEngine(0)(0)` определяет ссылку на открытую копию текущей базы данных. Функция **CurrentDb** позволяет создать несколько переменных типа **Database**, указывающих на текущую базу данных. В Microsoft Access по-прежнему поддерживается синтаксис `DBEngine(0)(0)`, однако, рекомендуется изменить существующие программы во избежание возможных конфликтов при работе в сети.

Объект DBEngine (Microsoft Access)

Microsoft Access содержит средства управления объектами доступа к данным (DAO) из других приложений с помощью программирования. Если Microsoft Access выполняется под управлением из другого приложения, например, Visual Basic или Microsoft Excel, то свойство **DBEngine** объекта **Application** Microsoft Access возвращает ссылку на объект **DBEngine**. После этого все объекты доступа к данным и семейства становятся доступными через объект **DBEngine**.

Объект Document, семейство Documents (Microsoft Access)

В дополнение к объектам **Document** (документ), определенным в ядре базы данных Microsoft Jet, в Microsoft Access определены следующие объекты **Document**.

Документ	Контейнер	Описываемые объекты
Form	Forms	Сохраненная форма
Macro	Scripts	Сохраненный макрос
Module	Modules	Сохраненный модуль
Report	Reports	Сохраненный отчет
SummaryInfo	Databases	Общие сведения о базе данных
UserDefined	Databases	Определяемые пользователем свойства

Объекты **Document**, определенные в Microsoft Access, представляют объекты базы данных Microsoft Access, а не объекты доступа к данным (DAO). Эти объекты **Document** передают в ядро базы данных Jet информацию об объектах Microsoft Access. Эта информация используется ядром Jet для обеспечения той же системы защиты объектов базы данных Microsoft Access, которая реализуется ядром для объектов доступа к данным.

Семейство **Documents** объекта **Container** содержит объекты **Document**, представляющие отдельные объекты, тип которых описывается в объекте **Container**. Например, объект **Container** с именем Forms содержит семейство **Documents**, в которое может входить объект **Document**, соответствующий форме **Заказы**.

В Microsoft Access определены два объекта **Document**, входящие в объект **Container** с именем **Databases** – **SummaryInfo** и **UserDefined**

Объект **Document SummaryInfo** обеспечивает доступ в программах к свойствам, содержащим общие сведения о документе, в том числе, **Title** (Название), **Subject** (Тема), **Author** (Автор), **Keywords** (Ключевые слова), **Comments** (Заметки), **Manager** (Должность), **Company** (Организация), **Category** (Группа) и **Hyperlink Base** (База гиперссылки). Значения всех этих свойств могут быть также заданы на вкладке **Документ** в диалоговом окне **Свойства: база данных**, которое открывается командой **Свойства** в меню **Файл**.

Для того чтобы задать значения этих свойств в программе Visual Basic, необходимо предварительно создать эти свойства и добавить их в семейство **Properties** объекта **Document SummaryInfo**, если эти свойства ранее не были определены в диалоговом окне **Свойства: база данных**. После создания этих свойств становятся возможными явные ссылки на них через семейство **Properties**. В следующем примере переменная `doc` является объектной переменной, представляющей объект **Document SummaryInfo**.

```
doc.Properties!Title = "Борей"
```

Для получения дополнительных сведений о создании и задании значений свойств, определяемых в Microsoft Access, следует обратиться к разделам справки, выводящимся в указателе справки для объекта **Property** и метода **CreateProperty**.

Объект **Document UserDefined** обеспечивает доступ в программах к свойствам, определяемым пользователем на вкладке **Прочие** в окне диалога **Свойства: база данных**. Пользователь имеет также возможность создать эти свойства в программе Visual Basic и добавить их в семейство **Properties** объекта **Document UserDefined**.

Примечание. Не следует путать свойства, определенные в семействе **Properties** документа **Database**, со свойствами, определенными в семействе **Properties** объекта **Database**. Создание свойств, определяемых пользователем, допускается в лбом из этих двух семейств **Properties**, однако, в окне диалога **Свойства: база данных** могут быть заданы только значения свойств, определенных для объектов **Document SummaryInfo** или **UserDefined**.

Объекты **Document** могут быть использованы для установления и обеспечения разрешения на доступ к отдельным объектам базы данных Microsoft Access. Для того чтобы установить разрешение на доступ к объекту **Document**, следует указать в свойстве **UserName** объекта **Document** имя существующего объекта **User** (пользователь) или **Group** (группа). После этого следует определить для объекта **Document** свойство **Permissions**.

Примечание. Не следует путать отдельные типы объектов **Container** с одноименными типами семейств. Объекты **Container** с именами **Forms** и **Reports** содержат семейство **Documents**, в которое входят отдельные объекты **Document**, представляющие каждую сохраненную форму или отчет. В семейства **Forms** и **Reports** включаются только открытые формы и отчеты.

Объект Field, семейство Fields (Microsoft Access)

В дополнение к свойствам, определенным в ядре базы данных Microsoft Jet, объект **Field** (поле), входящий в семейство **Fields** объекта **QueryDef** (запрос) или **TableDef** (таблица), может также содержать следующие свойства, определяемые в приложении Microsoft Access.

Более подробно о чтении и задании значений этих свойств см. в разделах справки о конкретном свойстве и объекте **Property** (свойство).

Подпись (Caption)	**Число десятичных знаков (DecimalPlaces)**
ColumnHidden	**Описание (Description)**
ColumnOrder	**Формат поля (Format)**
Ширина столбцов (ColumnWidths)	**Маска ввода (InputMask)**

Объект Group, семейство Groups (Microsoft Access)

Объекты **Group** (группа) создаются при создании и поддержании системы различных разрешений на доступ к объектам базы данных Microsoft Access и объектам доступа к данным. Например, группы используются при организации системы защиты форм, отчетов, макросов и модулей.

Объект **Group** имеет свойство **Name**, которое используется при указании разрешений на доступ к объекту **Container** или **Document**. Например, имя группы, указанное в свойстве **Name** объекта **Group**, можно присвоить свойству **UserName** объекта **Container** или **Document**. После этого следует задать значение свойства **Permissions** объекта **Container** и **Document**, определяющие права группы пользователей, указанной в свойстве **UserName**. Чтение значения свойства **Permissions** позволяет определить текущие права данной группы пользователей.

Объект Property, семейство Properties (Microsoft Access)

Целый ряд свойств является определенными в Microsoft Access. В программах Visual Basic эти свойства представляются с помощь объектных переменных, каждая из которых связана с объектом **Property** (свойство), входящим в семейство **Properties**.

Свойства, применимые к объектам доступа к данным

- Встроенные свойства, определенные в ядре базы данных Microsoft Jet для каждого объекта доступа к данным (DAO).
- Определяемые пользователем свойства, которые добавляются в некоторые объекты доступа к данным. Такими объектами являются объекты **Database** (база данных), **Index** (индекс), **QueryDef** (запрос) и **TableDef** (таблица), а также объекты **Field** (поле), входящие в семейство **Fields** объекта **QueryDef** или **TableDef**.
- Некоторые свойства, определенные в Microsoft Access, являются применимыми к объектам доступа к данным. Значения таких свойств в общем случае могут быть заданы как в окне Microsoft Access, так и в программе Visual Basic. Ядро базы данных Jet не распознает такие свойства без специального создания соответствующего объекта **Property** (свойство) и его добавления в семейство **Properties**. Объектами доступа к данным, к которым применятся такие свойства, являются объекты **QueryDef** и **TableDef**, а также объекты **Field**, входящие в семейство **Fields** объекта **QueryDef** или **TableDef**. Список этих свойств, определенных в Microsoft Access, можно найти в разделах справки, посвященных объектам **TableDef**, **QueryDef** и **Field**.

Имеется ряд отличий свойств, определенных в Microsoft Access, которые применятся к объектам доступа к данным, от свойств, определенных в ядре базы данных Jet.

Для ссылки на определяемое пользователем свойство или на свойство, определенное в Microsoft Access, необходимо в явном виде сослаться на семейство **Properties**. Быстрее всего выполнятся ссылки на свойства Microsoft Access, использующие следующий синтаксис:

объект.**Properties!***имя*

В этом примере *объект* является объектом доступа к данным (DAO), а *имя* – имя свойства Microsoft Access.

Допускается также следующий синтаксис, однако, такие конструкции выполнятся медленнее:

объект.**Properties("***имя***")**

В отличие от этого, на свойства, определенные в ядре базы данных Jet, допускаются прямые ссылки вида *объект*.*имя*.

При первом указании значения свойства, определенного в Microsoft Access, необходимо сначала создать объект **Property** с помощь метода **CreateProperty**. Например, свойство **Подпись (Caption)** объекта **Field** является свойством, определенным в Microsoft Access. Если значение свойства **Подпись (Caption)** не было задано в режиме конструктора таблицы, а задается впервые в программе Visual Basic, пользователь должен сначала создать это свойство с помощь метода **CreateProperty** и добавить его в семейство **Properties**, а лишь затем задать его значение.

Определенные в Microsoft Access свойства автоматически добавляются в семейство **Properties** при первом задании значения этого свойства в окне Microsoft Access, поэтому нет необходимости создавать и добавлять в семейство программным образом те свойства, значения которых были заданы через интерфейс пользователя. Например, значение свойства **Подпись (Caption)** можно определить в режиме таблицы с помощь команды **Шрифт** из меню **Формат**. Данное свойство будет автоматически включено в семейство **Properties** объекта **TableDef**, представляющего эту таблицу.

До задания значения свойства, определенного в Microsoft Access, либо в режиме таблицы, либо

в программе Visual Basic, это свойство не будет включено в семейство **Properties**. Если значение данного свойства задается в процедуре Visual Basic, необходимо включить в эту процедуру блок обработки ошибок, который будет проверять, существует ли это свойство в семействе **Properties**, и если нет, то добавлять данное свойство в семейство.

Свойства, применимые к объектам Microsoft Access

Аналогично объектам доступа к данным (DAO) каждый объект Microsoft Access содержит семейство **Properties**, в которое входят встроенные объекты **Property**. Например, объекты **Property**, соответствующие формам, являются компонентами семейства **Properties** объекта **Form**.

Допускается также создание определяемых пользователем свойств для объектов Microsoft Access. Например, можно создать свойство с именем TextType, которое будет применяться к элементу управления-полю.

В отличие объектов **Property** (свойств) объектов доступа к данным, объекты **Property**, входящие в семейство **Properties** объектов **Form**, **Report** и **Control**, не имеют свойства **Inherited**.

Допускается нумерация объектов **Property**, входящих в семейство **Properties** объектов **Form**, **Report** и **Control**. Однако представляющие Microsoft Access объекты **Application** и **Screen** имеют семейства **Properties**, компоненты которого не могут быть перенумерованы. Кроме того, свойства этих объектов являются доступными только для чтения.

Объект QueryDef, семейство QueryDefs (Microsoft Access)

В дополнение к свойствам, определенным в ядре базы данных Microsoft Jet, объект **QueryDef** (запрос) может также содержать следующие свойства, определенные в приложении Microsoft Access. Подробнее о чтении и задании значений таких свойств см. разделы справки для отдельных свойств и для объекта **Property**.

DatasheetFontItalic
DatasheetFontHeight
DatasheetFontName
DatasheetFontUnderline
DatasheetFontWeight
Описание (Description)
FrozenColumns
Таблица сообщений (LogMessages)
Блокировка записей (RecordLocks)
RowHeight
ShowGrid
UseTransaction

Объект TableDef, семейство TableDefs (Microsoft Access)

В дополнение к свойствам, определенным в ядре базы данных Microsoft Jet, объект **TableDef** (таблица) может также содержать следующие свойства, определенные в приложении Microsoft Access. Подробнее о чтении и задании значений таких свойств см. разделы справки для отдельных свойств и для объекта **Property**.

DatasheetFontHeight
DatasheetFontItalic
DatasheetFontName
DatasheetFontUnderline
DatasheetFontWeight
Описание (Description)
FrozenColumns
RowHeight
ShowGrid

Объект User, семейство Users (Microsoft Access)

При создании и поддержании системы различных разрешений на доступ к объектам базы данных Microsoft Access и объектам доступа к данным существует возможность создавать объекты **User** (пользователь). Например, имена пользователей используются при организации системы защиты форм, отчетов, макросов и модулей.

Объект **User** имеет свойство **Name,** которое используется при указании разрешений на объект **Container** или **Document** . Например, имя пользователя, указанное в свойстве **Name** объекта **User**, можно присвоить свойству **UserName** объекта **Container** или **Document**. После этого следует задать значение свойства **Permissions** объекта **Container** и **Document**, определяющее права пользователя, указанного в свойстве **UserName**. Чтение значения свойства **Permissions** позволяет определить текущие разрешения на доступ данного пользователя.

Объект Container, семейство Containers (Microsoft Access), пример

В данном примере программистам представляется разрешение на полный доступ на каждый модуль в базе данных, а все остальные пользователи получают разрешение только на чтение модулей:

```
Sub SetModulePermissions()
   Dim dbs As Database, wsp As Workspace, ctr As Container
   Dim grp As Group

   ' Возвращает ссылку на используемую по умолчанию рабочую область.
   Set wsp = DBEngine.Workspaces(0)
   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Возвращает ссылку на семейство Modules.
   Set ctr = dbs.Containers!Modules
   wsp.Groups.Refresh
   For Each grp In wsp.Groups
      ctr.UserName = grp.Name
      If ctr.UserName = "Программисты" Then
         ctr.Permissions = ctr.Permissions Or dbSecFullAccess
      Else
         ctr.Permissions = ctr.Permissions Or acSecModReadDef
      End If
   Next grp
   Set dbs = Nothing
End Sub
```

Объект Database, семейство Databases (Microsoft Access), пример

В данном примере демонстрируются три способа возврата объекта **Database** в Microsoft Access. Сначала процедура возвращает объект **Database**, представляющий текущую базу данных, открытую в окне Microsoft Access. Затем в процедуре создается и сохраняется на диске другая база данных Newdb.mdb. После этого открывается существующая база данных с именем Another.mdb. И наконец, выводится перечень всех объектов **Database**, содержащихся в семействе **Databases**.

```
Sub ReferenceDatabases()
   Dim wsp As Workspace
   Dim dbsCurrent As Database, dbsNew As Database
   Dim dbsAnother As Database, dbs As Database

   ' Возвращает ссылку на текущую базу данных.
   Set dbsCurrent = CurrentDb
   ' Возвращает ссылку на используемую по умолчанию рабочую область.
   Set wsp = DBEngine.Workspaces(0)
   ' Создает новый объект Database.
   Set dbsNew = wsp.CreateDatabase("Newdb.mdb", dbLangGeneral)
   ' Открывает другую (не текущую) базу данных.
   Set dbsAnother = wsp.OpenDatabase("Another.mdb")
   ' Выводит перечень имен баз данных.
   For Each dbs in wsp.Databases
      Debug.Print dbs.Name
   Next dbs
   For Each dbs In wsp.Databases
      Set dbs = Nothing
   Next dbs
   Set wsp = Nothing
End Sub
```

Объект Field, семейство Fields (Microsoft Access), пример

В данном примере создается новый объект **Field**, задаются значения некоторых его свойств, после чего объект добавляется в семейство **Fields** объекта **TableDef**. После этого в процедуре выводится перечень всех полей в семействе **Fields** объекта **TableDef**.

```
Sub NewField()
   Dim dbs As Database, tdf As TableDef
   Dim fld As Field

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Возвращает ссылку на таблицу «Сотрудники».
   Set tdf = dbs.TableDefs!Сотрудники
   ' Создает новый объект Field.
   Set fld = tdf.CreateField("Страховка")
   ' Задает значения свойств Type и Size объекта Field.
   fld.Type = dbText
   fld.Size = 11
   ' Добавляет поле в семейство.
   tdf.Fields.Append fld
   ' Выводит перечень всех полей в семействе Fields объекта TableDef.
   For Each fld in tdf.Fields
      Debug.Print fld.Name
   Next fld
   Set dbs = Nothing
End Sub
```

Объект Group, семейство Groups (Microsoft Access), пример

В данном примере создается новый объект **User**, после чего объект добавляется в семейство **Users** объекта **Workspace**. Затем создается новый объект **Group**, который добавляется в семейство **Groups** объекта **Workspace**. Новый объект **Group** добавляется также в семейство **Groups** объекта **User**. Далее новой группе представляется разрешение на изменение и удаление модулей.

Обратите внимание, что для включения пользователей в группу необходимо либо добавить объект **User** в семейство **Users** объекта **Group**, либо добавить объект **Group** в семейство **Groups** объекта **User**. Каждый из этих приемов обеспечивает включение указанного пользователя в указанную группу.

Примечание. При программировании разрешений следует избегать включения в программу пароля и идентификатора пользователя. Следующий пример является лишь иллюстрацией.

```
Sub NewModulesGroup()
   Dim wsp As Workspace, dbs As Database
   Dim usr As User, grp As Group, grpMember As Group
   Dim ctr As Container

   ' Возвращает ссылку на используемую по умолчанию рабочую область.
   Set wsp = DBEngine.Workspaces(0)
   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Создает объект User и добавляет его в семейство Users объекта
Workspace.
   Set usr = wsp.CreateUser("Барков", "123abc789xyz", "Пароль")
   wsp.Users.Append usr
   ' Создает объект Group и добавляет его в семейство Groups объекта
Workspace.
   Set grp = wsp.CreateGroup("Программисты", "321xyz987abc")
   wsp.Groups.Append grp
   ' Добавляет объект Group в семейство Groups объекта User.
   Set grpMember = usr.CreateGroup("Программисты")
   usr.Groups.Append grpMember
   ' Обновляет семейство Groups объекта User.
   usr.Groups.Refresh
   ' Возвращает объект Container.
   Set ctr = dbs.Containers!Modules
   ' Задает значение свойства UserName объекта Container.
   ctr.UserName = grpMember.Name
   'Определяет разрешение группы на изменение и удаление всех модулей.
   ctr.Permissions = ctr.Permissions Or acSecModWriteDef
   Set dbs = Nothing
   Set wsp = Nothing
End Sub
```

Объект DBEngine (Microsoft Access), пример

В данном примере выводится перечень свойств объекта **DBEngine**:

```
Sub EngineProperties()
   Dim prp As Property

   For Each prp In DBEngine.Properties
      Debug.Print prp.Name
   Next prp
End Sub
```

Объект Document, семейство Documents (Microsoft Access), пример

В данном примере выводятся имена всех объектов **Document** типа Form в текущей базе данных и даты последнего изменения каждой формы:

```
Sub DocumentModified()
   Dim dbs As Database, ctr As Container, doc As Document

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Возвращает ссылку на контейнер Forms.
   Set ctr = dbs.Containers!Forms
   ' Перечень содержимого семейства Documents в контейнере Forms.
   For Each doc In ctr.Documents
      ' Выводит имя объекта Document и значение свойства LastUpdated.
      Debug.Print doc.Name; "      "; doc.LastUpdated
   Next doc
   Set dbs = Nothing
End Sub
```

Динамический объект Recordset (Microsoft Access), пример

В данном примере создается объект **Recordset** типа динамического набора записей и проверяется значение его свойства **Updatable**:

```
Sub Новые_сотрудники()
   Dim dbs As Database, rst As Recordset
   Dim strSQL As String

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Открывает набор записей для таблицы «Сотрудники».
   Set rst = dbs.OpenRecordset("Сотрудники", dbOpenDynaset)
   Debug.Print rst.Updatable
   rst.Close
   Set dbs = Nothing
End Sub
```

Объект Error, семейство Errors (Microsoft Access), пример

В данном примере создается ошибка при попытке открыть объект **Recordset** для несуществующей таблицы «Студенты». Информация сохраняется в объекте доступа к данным **Error** и в объекте Visual Basic **Err**. В процедуре выводятся значения свойств **Description**, **Source** и **Number** объекта **Error**. Затем выводятся значения соответствующих свойств объекта **Err**.

Отметим, что последний объект **Error** в семействе **Errors** должен всегда определять ту же ошибку, что и объект **Err**. Т. е. значение **Err.Number** должно быть равно **Errors.Count** – 1. Если этого не происходит, то, вероятно, информация в семействе **Errors** является устаревшей. Для того чтобы обеспечить заполнение семейства **Errors** информацией о последних ошибках, следует вызвать метод **Refresh**.

```
Sub CheckError()
   Dim dbs As Database, tdf As TableDef, rst As Recordset
   ' Описывает объектную переменную типа Error
   ' для перечисления компонентов семейства Errors.
   Dim errX As Error

   ' Задает игнорирование ошибок.
   On Error Resume Next
   ' Очищает объект Err.
   Err.Clear
   ' Обновляет семейство Errors.
   Errors.Refresh
   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Попытка открыть объект Recordset для несуществующей таблицы.
   Set rst = dbs.OpenRecordset("Студенты")
   Debug.Print " ОДД Error:"
   ' Выводит число ошибок в семействе Errors.
   Debug.Print ">>>Число ошибок: "; Errors.Count
   ' Проверка ключевых свойств объектов в семействе Errors.
   For Each errX In DBEngine.Errors
      Debug.Print errX.Description
      Debug.Print errX.Source
      Debug.Print errX.Number
   Next errX
   Debug.Print
   Debug.Print "Объект VBA Err:"
   ' Вывод соответствующих свойств объекта Err.
   Debug.Print Err.Description
   Debug.Print Err.Source
   Debug.Print Err.Number
   Set dbs = Nothing
End Sub
```

Объект Index, семейство Indexes (Microsoft Access), пример

В данном примере создается новый индекс для таблицы «Сотрудники»:

```
Sub NewIndex()
   Dim dbs As Database, tdf As TableDef, idx As Index
   Dim fld1 As Field, fld2 As Field

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Возвращает ссылку на таблицу «Сотрудники».
   Set tdf = dbs.TableDefs!Сотрудники
   Set idx = tdf.CreateIndex("ИндексФамилия")
   Set fld1 = idx.CreateField("Фамилия")
   Set fld2 = idx.CreateField("Имя")
   idx.Fields.Append fld1
   idx.Fields.Append fld2
   idx.Required = True
   tdf.Indexes.Append idx
   Set dbs = Nothing
End Sub
```

Объект Parameter, семейство Parameters (Microsoft Access), пример

В данном примере создается новый запрос с параметрами и задаются значения параметров:

```
Sub NewParameterQuery()
   Dim dbs As Database, qdf As QueryDef, rst As Recordset
   Dim prm As Parameter, strSQL As String

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Создает строку SQL.
   strSQL = "PARAMETERS [Первая дата заказа] DateTime, " _
      & "[Последняя дата заказа] DateTime; SELECT * FROM Заказы " & _
      "WHERE (ДатаРазмещения Between[Первая дата заказа] " _
      & "And [Последняя дата заказа]);"
   ' Создает новый объект QueryDef.
   Set qdf = dbs.CreateQueryDef("ЗапросСПараметрами", strSQL)
   ' Задает значения параметров.
   qdf.Parameters![Первая дата заказа] = #4-1-95#
   qdf.Parameters![Последняя дата заказа] = #4-30-95#
   ' Открывает объект Recordset для объекта QueryDef.
   Set rst = qdf.OpenRecordset
   rst.MoveLast
   MsgBox "Получено " & rst.RecordCount & " записей."
   rst.Close
   Set dbs = Nothing
End Sub
```

Объект Property, семейство Properties (Microsoft Access), пример

В следующем примере создается свойство, которое является определенным в Microsoft Access, но применяется к объектам доступа к данным. Поскольку ядро базы данных Microsoft Jet не распознает свойства, определенные в Microsoft Access, необходимо при первом определении значения этого свойства создать новый объект **Property** и добавить его в семейство **Properties**.

Аналогично создается и определяемое пользователем свойство, применяемое к объекту Microsoft Access или объекту доступа к данным.

Отметим, что при создании свойства пользователь должен указать правильную константу для аргумента *тип.* Если неясно, какой тип данных следует указать, обратитесь к справочной системе за справкой о конкретном свойстве.

```
Function SetAccessProperty(obj As Object, strName As String, _
      intType As Integer, varSetting As Variant) As Boolean
   Dim prp As Property
   Const conPropNotFound As Integer = 3270

   On Error GoTo ErrorSetAccessProperty
   ' Явное обращение к семейству Properties.
   obj.Properties(strName) = varSetting
   obj.Properties.Refresh
   SetAccessProperty = True

ExitSetAccessProperty:
   Exit Function

ErrorSetAccessProperty:
   If Err = conPropNotFound Then
   ' Создает свойство, определяет его тип и задает начальное значение.
      Set prp = obj.CreateProperty(strName, intType, varSetting)
   ' Добавляет объект Property в семейство Properties.
      obj.Properties.Append prp
      obj.Properties.Refresh
      SetAccessProperty = True
      Resume ExitSetAccessProperty
   Else
      MsgBox Err & ": " & vbCrLf & Err.Description
      SetAccessProperty = False
      Resume ExitSetAccessProperty
   End If
End Function
```

Вызов вышеописанной функции выполняется, например, так:

```
Sub CallPropertySet()
   Dim dbs As Database, tdf As TableDef
   Dim blnReturn As Boolean

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Возвращает ссылку на таблицу «Сотрудники».
   Set tdf = dbs.TableDefs!Сотрудники
   ' Вызывает функцию SetAccessProperty.
   blnReturn = SetAccessProperty(tdf, _
      "DatasheetFontItalic", dbBoolean, True)
   ' Вычисляет возвращаемое значение.
```

```
   If blnReturn = True Then
      Debug.Print "Свойство успешно установлено."
   Else
      Debug.Print "Установка свойства не удалась."
   End If
End Sub
```

Объект QueryDef, семейство QueryDefs (Microsoft Access), пример

В данном примере проверяется наличие запроса «Новые сотрудники» в текущей базе данных, и если запрос найден, он удаляется из семейства **QueryDefs**. Затем в процедуре создается новый объект **QueryDef**, который открывается в режиме таблицы.

```
Sub NewQuery()
   Dim dbs As Database, qdf As QueryDef
   Dim strSQL As String

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Обновляет семейство QueryDefs.
   dbs.QueryDefs.Refresh
   ' Если запрос "Новые сотрудники" существует, он удаляется.
   For Each qdf in dbs.QueryDefs
      If qdf.Name = "Новые сотрудники" Then
         dbs.QueryDefs.Delete qdf.Name
      End If
   Next qdf
   ' Создает строку SQL, отбирающую сотрудников, нанятых после 1-1-94.
   strSQL = "SELECT * FROM Сотрудники WHERE ДатаНайма >= #1-1-94#;"
   ' Создает новый объект QueryDef.
   Set qdf = dbs.CreateQueryDef("Новые сотрудники", strSQL)
   ' Открывает запрос в режиме таблицы.
   DoCmd.OpenQuery qdf.Name
   Set dbs = Nothing
End Sub
```

Объект Recordset, семейство Recordsets (Microsoft Access), пример

В данном примере открывается объект **Recordset** типа таблицы, объект **Recordset** типа динамического набора записей и объект **Recordset** типа статического набора записей. Затем выводятся значения свойства **Updatable** объектов **Recordset**.

```
Sub NewRecordsets()
   Dim dbs As Database, rst As Recordset
   Dim rstEmployees As Recordset, rstOrders As Recordset
   Dim rstProducts As Recordset, strSQL As String

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Создает табличный объект Recordset.
   Set rstEmployees = dbs.OpenRecordset("Сотрудники", dbOpenTable)
   ' Создает динамический объект Recordset.
   Set rstOrders = dbs.OpenRecordset("Сотрудники", dbOpenDynaset)
   ' Создает статический объект Recordset.
   Set rstProducts = dbs.OpenRecordset("Товары", dbOpenSnapshot)
   ' Выводит значение свойства Updatable каждого объекта Recordset.
   For Each rst In dbs.Recordsets
      Debug.Print rst.Name; "   "; rst.Updatable
   Next rst
   ' Освобождftn все объектныt переменныt.
   rstEmployees.Close
   rstOrders.Close
   rstProducts.Close
   Set dbs = Nothing
End Sub
```

Объект Relation, семейство Relations (Microsoft Access), пример

В данном примере удаляется существующая связь между таблицами «Сотрудники» и «Заказы», а затем она создается заново с помощью нового объекта **Relation**.

```
Sub NewRelation()
   Dim dbs As Database
   Dim fld As Field, rel As Relation

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Ищет существующую связь СотрудникиЗаказы.
   For Each rel In dbs.Relations
      If rel.TABLE = "Сотрудники" And rel.ForeignTable = "Заказы" Then
         ' Выводит сообщение для пользователя перед удалением связи.
         If MsgBox(rel.Name & " уже существует. " & vbCrLf _
            & "Эта связь будет удалена и затем снова создана.", vbOK) =
vbOK Then
            dbs.Relations.Delete rel.Name
         ' Завершает процедуру, если пользователь нажал кнопку «Отмена».
         Else
            Exit Sub
         End If
      End If
   Next rel
   ' Создает новую связь и задает ее свойства.
   Set rel = dbs.CreateRelation("СотрудникиЗаказы", "Сотрудники", "Заказы")
   ' Задает атрибуты объекта Relation,
   ' обеспечивающие целостность данных.
   rel.Attributes = dbRelationDeleteCascade + dbRelationUpdateCascade
   ' Создает поле в семействе Fields объекта Relation.
   Set fld = rel.CreateField("КодСотрудника")
   ' Указывает имя поля внешнего ключа.
   fld.ForeignName = "КодСотрудника"
   ' Добавляет поле в объект Relation и объект Relation в базу данных.
   rel.Fields.Append fld
   dbs.Relations.Append rel
   MsgBox "Связь '" & rel.Name & "' создана."
   Set dbs = Nothing
End Sub
```

Статический объект Recordset (Microsoft Access), пример

В данном примере с помощь инструкции SQL создается статический объект **Recordset**, после чего выводится значение свойства **Updatable** этого объекта. Поскольку объект **Recordset** типа статического набора записей по определению является необновляемым, данное свойство всегда имеет значение **False** (0).

```
Sub LongTermEmployees()
   Dim dbs As Database, qdf As QueryDef, rst As Recordset
   Dim strSQL As String

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Создает строку SQL.
   strSQL = "SELECT * FROM Сотрудники WHERE ДатаНайма <= #1-1-94#;"
   ' Создает статический объект Recordset.
   Set rst = dbs.OpenRecordset(strSQL, dbOpenSnapshot)
   Debug.Print rst.Updatable
   rst.Close
   Set dbs = Nothing
End Sub
```

Табличный объект Recordset (Microsoft Access), пример

В данном примере открывается объект **Recordset** типа таблицы, и затем в нем ищется указанная запись:

```
Sub FindEmployee()
   Dim dbs As Database, tdf As TableDef
   Dim rst As Recordset, idx As Index
   Dim fldLastName As Field, fldFirstName As Field

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Возвращает ссылку на таблицу «Сотрудники».
   Set tdf = dbs.TableDefs!Сотрудники
   ' Создает новый индекс.
   Set idx = tdf.CreateIndex("ФИО")
   ' Создает и добавляет индексные поля.
   Set fldLastName = idx.CreateField("Фамилия", dbText)
   Set fldFirstName = idx.CreateField("Имя", dbText)
   idx.Fields.Append fldLastName
   idx.Fields.Append fldFirstName
   ' Добавляет индексный объект.
   tdf.Indexes.Append idx
   ' Открывает табличный объект Recordset.
   Set rst = dbs.OpenRecordset("Сотрудники")
   ' Делает новый индекс текущим.
   rst.Index = idx.Name
   ' Задает искомую запись.
   rst.Seek "=", "Иванов", "Андрей"
   If rst.NoMatch Then
      Debug.Print "Поиск не удался."
   Else
      Debug.Print "Успешный поиск."
   End If
   rst.close
   Set dbs = Nothing
End Sub
```

Объект TableDef, семейство TableDefs (Microsoft Access), пример

В данном примере создается новый объект **TableDef**, описывается поле в этом объекте, после чего объект добавляется в семейство **TableDefs** в текущей базе данных:

```
Sub NewTable()
   Dim dbs As Database, tdf As TableDef, fld As Field

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Создает новый объект TableDef.
   Set tdf = dbs.CreateTableDef("Контакты")
   ' Создает новый объект Field.
   Set fld = tdf.CreateField("Имя", dbText, 30)
   ' Добавляет новый объект.
   tdf.Fields.Append fld
   dbs.TableDefs.Append tdf
   Set dbs = Nothing
End Sub
```

Объект User, семейство Users (Microsoft Access), пример

В данном примере создается новый объект **User**, после чего объект добавляется в семейство **Users** объекта **Workspace**. Затем создается новый объект **Group**, который добавляется в семейство **Groups** объекта **Workspace**. Новый объект **User** добавляется также в семейство **Users** объекта **Group**. Новому пользователю дается разрешение на чтение данных из таблиц.

Отметим, что для включения пользователей в группы необходимо либо добавить объект **User** в семейство **Users** объекта **Group**, либо добавить объект **Group** в семейство **Groups** объекта **User**. В любом из этих случаев указанный пользователь будет включен в указанную группу.

Примечание. При программировании разрешений следует избегать включения в программу пароля и идентификатора пользователя. Следующий пример является лишь иллюстрацией.

```
Sub NewTablesUser()
   Dim wsp As Workspace, dbs As Database
   Dim usr As User, grp As Group, usrMember As User
   Dim ctr As Container, doc As Document

   ' Возвращает ссылку на используемую по умолчанию рабочую область.
   Set wsp = DBEngine.Workspaces(0)
   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Создает объект User и добавляет его в семейство Users объекта
Workspace.
   Set usr = wsp.CreateUser("Вася", "123abc456DEF", "Пароль")
   wsp.Users.Append usr
   ' Создает объект Group и добавляет его в семейство Groups объекта
Workspace.
   Set grp = wsp.CreateGroup("Сантехники", "321xyz654EFD")
   wsp.Groups.Append grp
   ' Добавляет новый объект User в семейство Users нового объекта Group.
   Set usrMember = grp.CreateUser("Вася")
   grp.Users.Append usrMember
   ' Обновляет семейство Users объекта Group.
   grp.Users.Refresh
   ' Возвращает объект Container.
   Set ctr = dbs.Containers!Tables
   ' Задает значение свойства UserName объекта Container.
   ctr.UserName = usrMember.Name
   ' Предоставляет новому пользователю разрешение на загрузку данных из
всех таблиц.
   ctr.Permissions = ctr.Permissions Or dbSecRetrieveData
   Set dbs = Nothing
   Set wsp = Nothing
End Sub
```

Свойство AllPermissions (Microsoft Access)

В дополнение к контейнерам**,** определенным в ядре базы данных Microsoft Jet, в Microsoft Access определены четыре типа объектов **Container**: Forms, Reports, Scripts и Modules. Отдельные объекты **Document** типа Form, Report, Script и Module включаются в семейства **Documents** соответствующих объектов **Container**. Для этих объектов **Container** и **Document** разрешение на доступ определяется с помощью свойства **AllPermissions** с использованием следующих констант.

Константа	**Права пользователя или группы**
acSecFrmRptReadDef	Открытие формы или отчета в режиме конструктора без внесения изменений.
AcSecFrmRptWriteDef	Изменение или удаление формы или отчета в режиме конструктора.
AcSecFrmRptExecute	Открытие формы в режиме формы или в режиме таблицы; печать или открытие отчета в режиме просмотра образца или в режиме предварительного просмотра.
acSecMacReadDef	Открытие окна макроса и просмотр макроса без внесения изменений.
AcSecMacWriteDef	Изменение или удаление макроса в окне макроса.
AcSecModReadDef	Открытие и просмотр модуля без внесения изменений.
AcSecModWriteDef	Изменение или удаление содержимого модуля.
AcSecMacExecute	Запуск макроса.

Свойство Bookmark (Microsoft Access)

При использовании свойства объекта доступа к данным (DAO) **Bookmark** в программе Visual Basic для определения закладки в объекте **Recordset** необходимо включить инструкцию **Option Compare Binary** в раздел описаний модуля. Свойство DAO **Bookmark** задает или возвращает закладку, представляющую из себя массив типа **Variant** данных, принадлежащих к типу **Byte**, поэтому для модуля необходимо выбрать двоичный способ сравнения строк. Если для проверки закладки используется способ сравнения текстовых строк, задающийся инструкцией **Option Compare Text**, или стандартная настройка **Option Compare Database**, то указатель текущей записи может быть установлен на неверную запись.

Примечание. Не следует путать данное свойство со свойством **Bookmark**, определенным в Microsoft Access, которое относится к объектам **Form** и сохраняет закладку в базовых таблицах или запросах формы. Эти два одноименные свойства существуют независимо и не мешают друг другу; можно одновременно определить закладки и для формы, и для объекта **Recordset**.

Свойство CacheStart (Microsoft Access)

При использовании свойства **CacheStart** в модуле Microsoft Access необходимо включить инструкцию **Option Compare Binary** в раздел описаний модуля. Свойство **CacheStart** задает или возвращает закладку, представляющую из себя массив типа **Variant** данных, принадлежащих к типу **Byte**, поэтому для модуля необходимо выбрать двоичный способ сравнения строк. Если для проверки закладки используется способ сравнения текстовых строк, задающийся инструкцией **Option Compare Text**, или стандартная настройка **Option Compare Database**, то указатель текущей записи может быть установлен на неверную запись.

Свойство «Значение по умолчанию» (DefaultValue) - (Microsoft Access)

Если в Microsoft Access свойству объектов доступа к данным (DAO) **Значение по умолчанию (DefaultValue)** для создания поля типа «Счетчик» присвоено значение **GenUniqueID( )**, необходимо также свойству **Attributes** присвоить значение **dbAutoIncrement**. Если этого не сделать, полю не будет присвоен тип «Счетчик», а при попытке просмотра таблицы в режиме таблицы будет выдано сообщение об ошибке.

В следующем примере показано создание поля с типом данных Счетчик. Объектная переменная `tdf` имеет тип **TableDef**, а объектная переменная `fld` – тип **Field**.

```
Set fld = tdf.CreateField("КодСотрудника", dbLong)
fld.DefaultValue = "GenUniqueID()"
fld.Attributes = dbAutoIncrField
tdf.Fields.Append fld
```

Если значение свойства **Значение по умолчанию (DefaultValue)** задается с помощью выражения, то в это выражение нельзя включать определяемые пользователем функции, статистические функции по подмножеству Microsoft Access, статистические функции SQL, функцию **CurrentUser**, функцию **Eval**, а также ссылки на запросы, формы или другие объекты **Field**.

Свойство Description (Microsoft Access)

Свойство объектов доступа к данным **Description**, определенное для объекта **Error**, возвращает строку, содержащую описание ошибки. Значение данного свойства задается только средствами Visual Basic.

Примечание. Не следует путать данное свойство со свойством **Описание (Description)** Microsoft Access, в котором определяется описание таблицы или запроса или их полей. Последнее свойство применяется к объекту **TableDef**, объекту **QueryDef** или объекту **Field** в семействе **Fields** объекта **TableDef** или **QueryDef**.

Свойство Filter (Microsoft Access)

Свойство объектов доступа к данным **Filter** задает фильтр для объекта **Recordset**. Значение данного свойства задается только средствами Visual Basic.

Примечание. Свойство Microsoft Access **Фильтр (Filter)** определяет обычный фильтр. Значение этого свойства можно задать в макросе, в программе Visual Basic или в окне свойств формы в режиме конструктора.

Свойства HelpContext, HelpFile (Microsoft Access)

Свойства объектов доступа к данным **HelpContext** и **HelpFile** определены для объекта **Error**. Значение данного свойства задается только средствами Visual Basic.

Примечание. Не следует путать данные свойства со свойствами Microsoft Access **Идентификатор справки (HelpContextID)** и **Файл справки (HelpFile)**. Эти свойства аналогичны свойствам объектов доступа к данным, но относятся только к формам, отчетам и элементам управления. Значения этих свойств можно задать в программе Visual Basic, в макросе или в окне свойств в режиме конструктора формы или режиме конструктора отчета.

Свойство Inherited (Microsoft Access)

В Microsoft Access определен ряд свойств, которые применятся к объектам доступа к данным. Поскольку эти свойства определены в Microsoft Access, они не распознаются автоматически ядром базы данных Microsoft Jet. Для того чтобы задать значение какого-либо из этих свойств в программе Visual Basic, необходимо сначала создать это свойство с помощью метода **CreateProperty**, а затем добавить его в семейство **Properties** соответствующего объекта. Более подробное описание см. в разделах, перечисленных в указателе справочной системы для объекта **Property**.

При создании объекта на основе другого объекта производный объект наследует свойства исходного объекта. Свойство **Inherited** позволяет определить, было ли создано конкретное свойство специально для объекта, или это свойство наследуется от другого объекта.

Предположим, например, что требуется определить свойство Microsoft Access **DatasheetFontName** для объекта **TableDef**. Если это свойство создается впервые, то необходимо создать соответствующий объект **Property** и добавить его в семейство **Properties** объекта **TableDef**. Свойство **Inherited** нового объекта **Property** получит значение **False** (0).

Если затем создается новый объект **QueryDef** (запрос), базовой таблицей которого является таблица, соответствующая объекту **TableDef**, то свойство **DatasheetFontName** будет включено в семейство **Properties** объекта **QueryDef**. Свойство **Inherited** объекта **Property**, соответствующего данному свойству, получит значение **True** (–1).

Свойство LastModified (Microsoft Access)

При использовании свойства **LastModified** в модуле Microsoft Access необходимо включить инструкцию **Option Compare Binary** в раздел описаний модуля. Свойство **LastModified** задает или возвращает закладку, представляющую из себя массив типа **Variant** данных, принадлежащих к типу **Byte,** поэтому для модуля необходимо выбрать двоичный способ сравнения строк. Если для проверки закладки используется способ сравнения текстовых строк, задающийся инструкцией **Option Compare Text**, или стандартная настройка **Option Compare Database**, то указатель текущей записи может быть установлен на неверную запись.

Свойство Permissions (Microsoft Access)

В дополнение к контейнерам, определенным в ядре базы данных Microsoft Jet, в Microsoft Access определены четыре типа объектов **Container** - объекты **Container** типа форма, отчет, макрос или модуль. Отдельные объекты **Document** типа форма, отчет, макрос или модуль включатся в семейства **Documents** соответствующих объектов **Container**. Для этих объектов **Container** и **Document** разрешение на доступ определяется с помощью свойства **Permissions** с использованием следующих констант.

Константа	Права пользователя или группы
acSecFrmRptReadDef	Открытие формы или отчета в режиме конструктора без внесения изменений.
acSecFrmRptWriteDef	Изменение или удаление формы или отчета в режиме конструктора.
acSecFrmRptExecute	Открытие формы в режиме формы или в режиме таблицы; печать или открытие отчета в режиме просмотра образца или в режиме предварительного просмотра.
acSecMacReadDef	Открытие окна макроса и просмотр макроса без внесения изменений.
acSecMacWriteDef	Изменение или удаление макроса в окне макроса.
acSecModReadDef	Открытие и просмотр модуля без внесения изменений.
acSecModWriteDef	Изменение или удаление содержимого модуля.
acSecMacExecute	Запуск макроса.

Свойство Primary (Microsoft Access)

При определении ключа таблицы в конструкторе таблиц Microsoft Access ключ автоматически используется в качестве основного индекса.

Свойство Size (Microsoft Access)

Свойство **Size** эквивалентно свойству **Размер поля (FieldSize)**, значение которого задается в режиме конструктора таблицы.. Задать значение свойства **Размер поля (FieldSize)** средствами Visual Basic невозможно; вместо этого следует использовать свойство **Size**.

В Microsoft Access поле Memo может содержать до 1,2 гигабайт данных. Однако в Microsoft Access можно отобразить в элементе управления в форме или отчете только первые 32 килобайта.

Свойство Source (Microsoft Access)

Свойство объектов доступа к данным **Source**, определенное для объекта **Error**, доступно только для чтения в программах Visual Basic.

Примечание. Не следует путать данное свойство со свойством **Source** Microsoft Access, определенным для запросов, в котором задается строка подключения к источнику и база данных, содержащая исходную таблицу или запрос.

Свойство Type (Microsoft Access)

В Microsoft Access допускается определение типа данных для поля таблицы в режиме конструктора таблицы, а также для параметра в диалоговом окне **Параметры запроса**. Эти действия эквивалентны определению значения свойства объектов доступа к данным **Type** для объекта **Field** или объекта **Parameter** в программе Visual Basic.

В следующей таблице перечислены константы, определяющие значение свойства **Type**, и соответствующие настройки свойств полей Microsoft Access и типов данных параметров, которые задаются в режиме конструктора таблицы или в диалоговом окне **Параметры запроса**.

При создании поля таблицы с типом данных числовой следует также в свойстве **Размер поля (FieldSize)** указать один из шести возможных числовых типов данных поля. По умолчанию, числовые поля получат размер «Длинное целое». Другими допустимыми размерами являются «Байт», «Целое», «С плавающей точкой (4 байт)», «С плавающей точкой (8 байт)» и «Код репликации».

Константа	Тип поля таблицы	Тип параметра запроса
dbBoolean	Логический	Логический
dbByte	Числовой (**Размер поля** = Байт)	Байт
dbCurrency	Денежный	Денежный
dbDate	Дата/время	Дата/время
dbDouble	Числовой (**Размер поля** = С плавающей точкой (8 байт))	С плавающей точкой
dbGUID	Числовой или Счетчик (**Размер поля** = Код репликации)	*Не поддерживается*
dbInteger	Числовой (**Размер поля** = Целое)	Целое
dbLong	Числовой (**Размер поля** = Длинное целое)	Длинное целое
	Счетчик (**Размер поля** = Длинное целое)	*Не поддерживается*
dbLongBinary	Поле объекта OLE	Поле объекта OLE
	Не поддерживается	Двоичный
dbMemo	Поле MEMO	Поле MEMO
dbSingle	Числовой (**Размер поля** = С плавающей точкой (4 байт))	С плавающей точкой
dbText	Текстовый	Текстовый
Не поддерживается	*Не поддерживается*	Значение

Примечание. Тип параметра «Значение» не соответствует ни одному из типов данных, определенных в ядре базы данных Microsoft Jet. Этот тип соответствует зарезервированному слову VALUE в языке SQL, которое используется при создании запросов с параметрами. В запросах Microsoft Access или запросах SQL ключевое слово VALUE может рассматриваться как допустимый синоним типа **Variant** Visual Basic.

Свойство «Условие на значение» (ValidationRule) - (Microsoft Access)

С помощью свойства **ValidationRule** определяют условия на значение для объектов **Field** (поле), **Recordset** (набор записей) или **TableDef** (таблица) в программах Visual Basic.

Кроме того, допускается задание условий на значение поля или элемента управления в интерфейсе пользователя Microsoft Access с помощью свойства **Условие на значение (ValidationRule)** в режиме конструктора таблицы. для поля и в режиме конструктора формы для элемента управления.

В Microsoft Access строковое выражение, задающее значение свойства **Условие на значение (ValidationRule)** объекта **Field**, не может содержать ссылки на определяемые пользователем функции, статистические функции по подмножеству., статистические функции SQL, функцию **CurrentUser**, функцию **Eval**, а также ссылки на запросы.

Свойство «Размер поля» (FieldSize) - (Microsoft Access)

Свойство объектов доступа к данным (DAO) **Размер поля (FieldSize)** отличается от свойства Microsoft Access **Размер поля (FieldSize)**, устанавливаемого в режиме конструктора таблицы. Свойство DAO **Размер поля (FieldSize)** возвращает число байт, занимаемое объектом **Field** (поле) типа Memo или объектом OLE.

В Microsoft Access свойство **Размер поля (FieldSize)** используется для ограничения размеров поля в таблице. Это свойство доступно только в режиме конструктора. В программе на Visual Basic для этого следует воспользоваться свойством **Size**.

Свойство Attributes (Microsoft Access)

Если значение свойства **Attributes** объекта **TableDef** равно **dbHiddenObject**, Microsoft Access сокроет объект **TableDef**. Объект **TableDef** будет существовать в семействе **TableDefs**, но не будет отображаться в окне базы данных, даже если установлен флажок **Скрытые объекты** в диалоговом окне **Параметры**, вызываемом из меню **Сервис**.

Чтобы создать объект **TableDef**, который может быть скрыт и отображен, следует присвоить свойству **Attributes** значение **dbSystemObject**. После этого объект **TableDef** может быть сделан видимым путем установки флажка **Системные объекты** в диалоговом окне **Параметры**.

Свойство Version (Microsoft Access)

Базы данных, созданные в Microsoft Access 97 и Microsoft Access 95 имеют одинаковый формат. Поэтому значение свойства **Version** объекта **Database** равно 3.0 для баз данных, созданных или преобразованных в формат Microsoft Access 97 или Microsoft Access 95. Если необходимо различать формат баз данных Microsoft Access 97 и Microsoft Access 95, для получения версии Microsoft Access, в которой была создана данная база данных, следует воспользоваться функцией **SysCmd**, указав в качестве параметра *действие* встроенную константу **acSysCmdAccessVer**. Для базы данных Microsoft Access 97 функция **SysCmd** вернет значение 8.0. Для базы данных Microsoft Access 95 - 7.0.

Свойство AbsolutePosition (Microsoft Access), пример

В следующем примере выполняется поиск заданной записи в динамическом объекте **Recordset** и определяется ее порядковый номер:

```
Sub ПорядковыйНомерЗаписи()
   Dim dbs As Database, rst As Recordset

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Возвращает ссылку на динамический объект Recordset.
   Set rst = dbs.OpenRecordset("SELECT * FROM Заказы WHERE " _
      & "ДатаНазначения >= #1-1-96#;")
   ' Находит первый заказ, отправляемый в Лондон.
   rst.FindFirst "ГородПолучателя = 'Лондон'"
   ' Возвращает порядковый номер этой записи.
   Debug.Print rst.AbsolutePosition
   rst.Close
   Set dbs = Nothing
End Sub
```

Свойство Пустые строки (Microsoft Access – Русский), пример

Свойство AllowZeroLength (Microsoft Access – Английский), пример

В следующем примере создается новый объект **Field** и его свойству **Пустые строки (AllowZeroLength)** присваивается значение **True** (–1):

```
Sub ZeroLengthField()
   Dim dbs As Database, tdf As TableDef, fld As Field

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Возвращает ссылку на таблицу «Сотрудники».
   Set tdf = dbs.TableDefs!Сотрудники
   ' Создает новое поле в таблице «Сотрудники».
   Set fld = tdf.CreateField("ИмяСупруга", dbText, 15)
   ' Разрешает в поле ввод пустых строк.
   fld.AllowZeroLength = True
   ' Добавляет объект Field.
   tdf.fields.Append fld
   Set dbs = Nothing
End Sub
```

Свойство AllPermissions (Microsoft Access), пример

В следующем примере проверяется значение свойства **AllPermissions** объекта **Container** с именем Forms и определяется, имеет ли заданный в свойстве **UserName** пользователь полный доступ к форме.

Следует отметить, что оператор **And** выполняет поразрядное сравнение для определения установленных разрешений.

```
Sub CheckAllPermissions()
   Dim dbs As Database, ctr As Container

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Возвращает ссылку на семейство Forms.
   Set ctr = dbs.Containers!Forms
   ' Проверяет, задано ли в свойстве AllPermissions разрешение на полный
доступ.
   If (ctr.AllPermissions And dbSecFullAccess = dbSecFullAccess) Then
      MsgBox "Пользователь " & ctr.UserName & " имеет разрешение на полный
доступ ко всем формам."
   End If
   Set dbs = Nothing
End Sub
```

Свойство Attributes (Microsoft Access), пример

В следующем примере проверяется значение свойства **Attributes** каждой таблицы текущей базы данных, а также отображаются имена системных и скрытых таблиц ядра базы данных Microsoft Jet.

Следует отметить, что оператор **And** выполняет поразрядное сравнение для определения установленных атрибутов.

```
Sub CheckAttributes()
   Dim dbs As Database, tdf As TableDef

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   For Each tdf In dbs.TableDefs
      ' Сравнивает значение свойства с заданными константами.
      If (tdf.Attributes And dbSystemObject) Or _
            (tdf.Attributes And dbHiddenObject) Then
         Debug.Print tdf.Name
      End If
   Next tdf
   Set dbs = Nothing
End Sub
```

Свойства BOF и EOF (Microsoft Access), пример

В следующем примере используется свойство **EOF** объекта **Recordset** для установки указателя текущей записи за последней записью объекта **Recordset.** Затем с помощью свойства **BOF** указатель текущей записи устанавливается перед первой записью этого объекта.

```
Sub EndsOfFile()
   Dim dbs As Database, rst As Recordset

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Открывает набор записей для таблицы «Заказы».
   Set rst = dbs.OpenRecordset("Заказы")
   ' Выполняет до окончания файла.
   Do Until rst.EOF
      ' Переходит к следующей записи.
      rst.MoveNext
      .
      .
      .
   Loop
   ' Переходит к последней записи.
   rst.MoveLast
   ' Выполняет до начала файла.
   Do Until rst.BOF
      rst.MovePrevious
      .
      .
      .
   Loop
   rst.Close
   Set dbs = Nothing
End Sub
```

Свойства Bookmark и Bookmarkable (Microsoft Access), пример

В следующем примере выполняется обработка всех записей таблицы «Сотрудники» и значение свойства **Bookmark** для каждой записи записывается в массив.

```
' Включите эту инструкцию в раздел описаний модуля.
Option Compare Binary

Sub RecordPositions()
   Dim dbs As Database, rst As Recordset, fld As Field
   Dim intI As Integer
   ' Описывает массив для хранения закладок.
   Dim varRecord() As Variant

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Открывает табличный объект Recordset.
   Set rst = dbs.OpenRecordset("Сотрудники")
   ' Возвращает ссылку на поле «Фамилия».
   Set fld = rst.Fields!Фамилия
   ' Изменяет верхнюю границу размерности массива до значения свойства
RecordCount.
   ReDim varRecord(0 To rst.RecordCount - 1)
   intI = 0
   ' Проверяет свойство Bookmarkable объекта Recordset.
   If rst.Bookmarkable Then
      Do Until rst.EOF
         ' Заполняет массив закладками.
         varRecord(intI) = rst.Bookmark
         ' Увеличивает счетчик.
         intI = intI + 1
         rst.MoveNext
      Loop
   End If
   rst.Close
   Set dbs = Nothing
End Sub
```

Свойство CollatingOrder (Microsoft Access), пример

В следующем примере для текущей базы данных проверяется значение свойства **CollatingOrder**:

```
Sub SetSortOrder()
   Dim dbs As Database

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Проверяет значение свойства CollatingOrder базы данных.
   Debug.Print dbs.CollatingOrder
   Set dbs = Nothing
End Sub
```

Свойства Connect и SourceTableName (Microsoft Access), пример

В следующем примере в указанной базе данных создается объект **TableDef**. Затем задаются значения его свойств **Connect** и **SourceTableName** и объект добавляется в семейство **TableDefs**.

```
Sub ConnectSource()
   Dim dbs As Database, tdf As TableDef

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Создает новый объект TableDef.
   Set tdf = dbs.CreateTableDef("АвторPDX")
   ' Устанавливает связь с таблицей Paradox «Авторы» в базе данных C:\PDX\
Публикации.
   tdf.Connect = "Paradox 4.X;Database=C:\PDX\Публикации"
   tdf.SourceTableName = "Авторы"
   dbs.TableDefs.Append tdf
   Set dbs = Nothing
End Sub
```

Свойства Container и Permissions (Microsoft Access), пример

В следующем примере определяется имя объекта **Container,** которому принадлежит объект **Document**, и разрешения на доступ устанавливаются в соответствии с указанной учетной записью группы. Перед установкой разрешений для объекта необходимо указать конкретного пользователя или учетную запись группы. В следующем примере предполагается, что существует группа «Продавцы».

В Microsoft Access помимо объектов ядра базы данных Microsoft Jet определен дополнительные объекты **Container**. Это объекты с именами Form, Report, Script и Module.

Оператор **Or** выполняет поразрядное сравнение для определения установленных атрибутов.

```
Sub SetDocPermissions(docUnknown As Document)
   Dim strContainerName As String

   ' Присваивает свойству UserName значение допустимой учетной записи
группы.
   docUnknown.UserName = "Продавцы"
   ' Получает значение свойства Container.
   strContainerName = docUnknown.Container.Name
   Select Case strContainerName
      Case "Forms"
         ' Задает разрешения для объекта Document с именем Form.
         docUnknown.Permissions = docUnknown.Permissions Or _
            acSecFrmRptWriteDef
      Case "Reports"
         ' Задает разрешения для объекта Document с именем Report.
         docUnknown.Permissions = docUnknown.Permissions Or _
            acSecFrmRptExecute
      Case "Scripts"
         ' Задает разрешения для объекта Document с именем Script.
         docUnknown.Permissions = docUnknown.Permissions Or _
            acSecMacWriteDef
      Case "Modules"
         ' Задает разрешения для объекта Document с именем Module.
         docUnknown.Permissions = docUnknown.Permissions Or _
            acSecModReadDef
      Case Else
         Exit Sub
   End Select
End Sub
```

Свойство Count (Microsoft Access), пример

В следующем примере выводятся имена и количество полей в таблице «Заказы»:

```
Sub CountFields()
   Dim dbs As Database, tdf As TableDef
   Dim fld As Field

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Возвращает ссылку на таблицу «Заказы».
   Set tdf = dbs.TableDefs!Заказы
   ' Подсчитать число полей в семействе Fields объекта TableDef.
   Debug.Print tdf.Fields.Count
   ' Выводит имена всех полей.
   For Each fld In tdf.Fields
      Debug.Print fld.Name
   Next fld
   Set dbs = Nothing
End Sub
```

Свойство DataUpdatable (Microsoft Access), пример

В следующем примере из одной и той же таблицы создаются два объекта **Recordset** – динамический объект **Recordset** и статический объект **Recordset**. Затем для каждого объекта **Recordset** проверяется значение свойства **DataUpdatable** поля «Фамилия». Значение свойства **DataUpdatable** для динамического объекта **Recordset** равно **True** (–1), а для статического объекта **Recordset** – **False** (0).

```
Sub CheckUpdatable()
   Dim dbs As Database
   Dim rstDynaset As Recordset, rstSnapshot As Recordset
   Dim fldDynaset As Field, fldSnapshot As Field

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Открывает динамический объект Recordset.
   Set rstDynaset = dbs.OpenRecordset("Сотрудники", dbOpenDynaset)
   ' Открывает статический объект Recordset.
   Set rstSnapshot = dbs.OpenRecordset("Сотрудники", dbOpenSnapshot)
   ' Получает объектные переменные типа Field, указывающие на поле в каждом
объекте Recordset.
   Set fldDynaset = rstDynaset.Fields!Фамилия
   Set fldSnapshot = rstSnapshot.Fields!Фамилия
   ' Получает текущую запись.
   rstDynaset.MoveFirst
   rstSnapshot.MoveFirst
   ' Выводит значение свойства DataUpdatable для каждого объекта Recordset.
   Debug.Print "DataUpdatable (динамический объект Recordset): "; _
      fldDynaset.DataUpdatable
   Debug.Print "DataUpdatable (статический объект Recordset): "; _
      fldSnapshot.DataUpdatable
   rstDynaset.Close
   rstSnapshot.Close
   Set dbs = Nothing
End Sub
```

Свойства DateCreated и LastUpdated (Microsoft Access), пример

В следующем примере перебираются все объекты **QueryDef** базы данных и выводятся те из них, которые созданы или изменены в течение последних шестидесяти дней:

```
Sub TimeUpdated()
   Dim dbs As Database, tdf As TableDef, qdf As QueryDef
   Dim strList As String, dteWeek As Date

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   dteWeek = Now - 60
   ' Перебирает все объекты семейства QueryDefs.
   For Each qdf In dbs.QueryDefs
      ' Анализирует значения свойств LastUpdated и DateCreated.
      If (qdf.LastUpdated > dteWeek) Or (qdf.DateCreated > dteWeek) Then
         ' Создает список.
         strList = strList & vbCrLf & qdf.Name
      End If
   Next qdf
   MsgBox strList
   Set dbs = Nothing
End Sub
```

Свойство «Значение по умолчанию» (DefaultValue) - (Microsoft Access), пример

В следующем примере создается новый объект **Field** и устанавливается его свойство **Значение по умолчанию (DefaultValue)**. Затем этот объект добавляется в семейство **Fields** таблицы «Сотрудники» семейства **TableDefs** базы данных.

```
Sub NewField()
   Dim dbs As Database, tdf As TableDef
   Dim fld As Field

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Возвращает ссылку на таблицу «Сотрудники».
   Set tdf = dbs.TableDefs!Сотрудники
   ' Создает объект Field.
   Set fld = tdf.CreateField("ДнейВОтпуске", dbText, 20)
   ' Задает значение свойства поля.
   fld.DefaultValue = "10"
   ' Добавляет поле fld в семейство Fields.
   tdf.Fields.Append fld
   Set dbs = Nothing
End Sub
```

Свойство DistinctCount (Microsoft Access), пример

В следующем примере выводится число уникальных значений в индексах по полям «КодЗаказа» и «ДатаРазмещения» таблицы «Заказы». Следует отметить, что свойство **DistinctCount** гарантированно возвращает число уникальных значений в индексе только сразу после создания индекса с помощью метода **CreateIndex**, а также после сжатия или преобразования базы данных помощью метода **CompactDatabase**.

```
Sub ПодсчетКлючей()
   Dim dbs As Database, tdf As TableDef
   Dim idx As Index, fldOrderID As Field, fldOrderDate As Field

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Возвращает ссылку на таблицу «Заказы».
   Set tdf = dbs.TableDefs!Заказы
   ' Создает новый индекс.
   Set idx = tdf.CreateIndex("ДатаРазмещения")
   ' Создает и добавляет индексные поля.
   Set fldOrderDate = idx.CreateField("ДатаРазмещения", dbDate)
   idx.Fields.Append fldOrderDate
   ' Добавляет новый индекс.
   tdf.Indexes.Append idx
   ' Обновляет семейство Indexes.
   tdf.Indexes.Refresh
   ' Выводит значение свойства DistinctCount для нового индекса.
   Debug.Print idx.DistinctCount
   Set dbs = Nothing
End Sub
```

Свойство «Фильтр» (Filter) - (Microsoft Access), пример

В следующем примере пользователь предлагается ввести название страны, по которой требуется установить фильтр, а затем задается значение свойства **Фильтр (Filter)** динамического объекта **Recordset**. Например, при вводе пользователем названия «Италия», отфильтрованный набор будет содержать только записи, значение поля «СтранаПолучателя» в которых равно «Италия».

Следует отметить, что сначала создается динамический объект **Recordset** и задается значение его свойства **Фильтр (Filter)**, а затем на его основе открывается второй динамический объект **Recordset**. Изменение значения свойства **Фильтр (Filter)** первого объекта **Recordset** в действительности не влияет на содержащиеся в нем записи, поэтому, чтобы увидеть результат работы фильтра, необходимо создать второй объект **Recordset**.

```
Sub SetRecordsetFilter()
   Dim dbs As Database, strInput As String
   Dim rstOrders As Recordset, rstFiltered As Recordset

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Запрашивает значение для фильтра.
   strInput = InputBox("Введите название страны, по которому следует
установить фильтр.")
   ' Создает динамический объект Recordset.
   Set rstOrders = dbs.OpenRecordset("Заказы", dbOpenDynaset)
   ' Задает условие отбора записей.
   rstOrders.Filter = "СтранаПолучателя = '" & strInput & "'"
   ' Создает динамический объект Recordset для отобранных записей.
   Set rstFiltered = rstOrders.OpenRecordset
   rstOrders.Close
   rstFiltered.Close
   Set dbs = Nothing
End Sub
```

Совет. В большинстве случает более эффективно создание второго объекта **Recordset** с заданными условиями за один шаг. Когда заранее известно, какой нужен фильтр, эффективнее использовать для создания набора записей инструкцию SQL. В следующем примере показано, как, создав всего один набор записей, достичь того же результата, что и в предыдущем примере:

```
Sub CreateRecordsetWithSQL()
   Dim dbs As Database, rst As Recordset
   Dim strInput As String

   Set dbs = CurrentDb
   strInput = InputBox("Введите название страны, по которому следует
установить фильтр.")
   Set rst = dbs.OpenRecordset("SELECT * FROM Заказы " _
      & "WHERE СтранаПолучателя = '" & strInput & "';")
   rst.MoveLast
   MsgBox "Набор записей содержит " & rst.RecordCount & " записей."
   rst.Close
   Set dbs = Nothing
End Sub
```

Свойство Foreign (Microsoft Access), пример

В следующем примере для каждого индекса в таблице «Заказы» выводится значение свойства **Foreign**:

```
Sub CheckForeign()
   Dim dbs As Database, tdf As TableDef, idx As Index

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Возвращает ссылку на таблицу «Заказы».
   Set tdf = dbs.TableDefs!Заказы
   ' Перебирает все объекты семейства Indexes таблицы.
   For Each idx In tdf.Indexes
      ' Выводит значение свойства Foreign.
      Debug.Print idx.Name; "        "; idx.Foreign
   Next idx
   Set dbs = Nothing
End Sub
```

Свойства ForeignName, ForeignTable и Table (Microsoft Access), пример

В следующем примере показано использование свойств **ForeignName**, **ForeignTable** и **Table** при создании отношения между двумя существующими таблицами – в данном случае «Сотрудники» (главная таблица) и «Заказы» (внешняя таблица с ключом) текущей базы данных. Поле «КодСотрудника» является ключевым полем в таблице «Сотрудники», а также внешним ключевым полем в таблице «Заказы». Отношение имеет тип «один ко многим» и обеспечивается целостность данных.

Следует отметить, что оператор **And** выполняет поразрядное сравнение для определения установленных разрешений.

```
Sub НовоеОтношение()
   Dim dbs As Database
   Dim fld As Field, rel As Relation

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Находит существующее отношение СотрудникиЗаказы.
   For Each rel In dbs.Relations
      If rel.Table = "Сотрудники" _
            And rel.ForeignTable = "Заказы" Then
         ' Предупреждает пользователя перед удалением отношения.
         If MsgBox(rel.Name & " уже существует. " & vbCrLf _
               & "Это отношение будет удалено, а затем вновь создано.", _
               vbOK) = vbOK Then
            dbs.Relations.Delete rel.Name
         ' Если пользователь отказался, завершает процедуру.
         Else
            Exit Sub
         End If
      End If
   Next rel
   ' Создает новое отношение и устанавливает значения его свойств.
   Set rel = dbs.CreateRelation("СотрудникиЗаказы")
   ' Задает значение свойства Table.
   rel.Table = "Сотрудники"
   ' Задает значение свойства ForeignTable.
   rel.ForeignTable = "Заказы"
   ' Задает атрибуты объекта Relation для обеспечения целостности данных.
   rel.Attributes = dbRelationDeleteCascade And dbRelationUpdateCascade
   ' Создает поле в семействе Fields объекта Relation.
   Set fld = rel.CreateField("КодСотрудника")
   ' Задает имя внешнего ключевого поля.
   fld.ForeignName = "КодСотрудника"
   ' Добавляет поле в объект Relation, а объект Relation в базы данных.
   rel.Fields.Append fld
   dbs.Relations.Append rel
   MsgBox "Отношение '" & rel.Name & "' создано."
   Set dbs = Nothing
End Sub
```

Свойства IgnoreNulls и Unique (Microsoft Access), пример

В следующем примере в объект **TableDef** добавляется новый индекс, а свойствам **IgnoreNulls** и **Unique** присваивается значение **True** (–1):

```
Sub НовыйИндекс()
   Dim dbs As Database, tdf As TableDef
   Dim idx As Index, fldID As Field, fldName As Field

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Возвращает ссылку на таблицу «Товары».
   Set tdf = dbs.TableDefs!Товары
   ' Возвращает объект Index, указывающий на новый индекс.
   Set idx = tdf.CreateIndex("КодИмя")
   ' Создает и добавляет индексное поле.
   Set fldID = idx.CreateField("КодТовара")
   Set fldName = idx.CreateField("Марка")
   idx.Fields.Append fldID
   idx.Fields.Append fldName
   ' Задает значения свойств индекса.
   idx.IgnoreNulls = True
   idx.Unique = True
   ' Добавляет новый объект Index в семейство Indexes.
   tdf.Indexes.Append idx
   Set dbs = Nothing
End Sub
```

Свойство Index (Microsoft Access), пример

В следующем примере в табличном объекте **Recordset** устанавливается индекс по ключевому полю. «PrimaryKey» является именем объекта **Index**, задаваемым по умолчанию, если индекс соответствует ключевому полю, установленному в режиме конструктора таблицы. Затем пользователю предлагается ввести значение и выполняется поиск этого значения в ключевом поле. Следует отметить, что перед выполнением над табличным объектом **Recordset** определенных операций, таких как метод **Seek**, следует задать текущий индекс.

```
Sub UsePrimaryKey()
   Dim dbs As Database, rst As Recordset
   Dim fld As Field, strInput As String

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Создает табличный объект Recordset.
   Set rst = dbs.OpenRecordset("Заказы", dbOpenTable)
   ' Устанавливает текущий индекс.
   rst.Index = "PrimaryKey"
   strInput = InputBox("Введите искомый КодЗаказа.")
   ' Находит запись.
   rst.Seek "=", strInput
   If Not rst.NoMatch Then
      For Each fld in rst.Fields
         Debug.Print fld.Name; "   "; fld.Value
      Next fld
   End If
   rst.Close
   Set dbs = Nothing
End Sub
```

Свойство Inherit (Microsoft Access), пример

В следующем примере свойству **Inherit** объекта **Container** с именем Forms присваивается значение **True** (–1), а затем в свойстве **Permissions** пользователю дается разрешение на полный доступ ко всем формам. Все новые объекты **Document** типа Form будут наследовать это разрешение. После этого в процедуре создается новый объект **Form** и выводятся разрешения для этого объекта.

Следует отметить, что объект **Container** с именем Forms отличается от семейства **Forms**. Объект **Container** с именем Forms включает все объекты **Document** с именем Form, существующие в базе данных; семейство **Forms** включает только открытые в настоящий момент формы.

Оператор **Or** выполняет поразрядное сравнение для определения установленных разрешений.

```
Sub FormPermissions()
   Dim dbs As Database, ctrForms As Container
   Dim docForm As Document, frmNew As Form

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Возвращает ссылку на семейство Forms.
   Set ctrForms = dbs.Containers!Forms
   ' Задает значение свойства Inherit.
   ctrForms.Inherit = True
   ' Указать, что разрешения должны наследоваться.
   ctrForms.Permissions = ctrForms.Permissions Or acSecFrmRptWriteDef
   ' Создает новую форму.
   Set frmNew = CreateForm()
   ' Сохраняет форму.
   DoCmd.Save , "ФормаЗаказа"
   ' Возвращает объект Document, связанный с новой формой.
   Set docForm = ctrForms.Documents!ФормаЗаказа
   ' Сравнивает разрешения для объектов Container и Document.
   If docForm.Permissions <> ctrForms.Permissions Then
      MsgBox "Ошибка!"
   Else
      MsgBox "Разрешения успешно унаследованы."
   End If
   Set dbs = Nothing
End Sub
```

Свойство Inherited (Microsoft Access), пример

В следующем примере в семействе **Properties** объекта **TableDef** создается объект **Property**, а затем на основе той же таблицы создается новый объект **QueryDef**. Объект **Property** автоматически существует в семействе **Properties** нового объекта **QueryDef**. Затем проверяется значение свойства **Inherited** объекта **Property** семейства **Properties** для объектов **TableDef** и **QueryDef**.

В этом примере свойство Microsoft Access **DatasheetFontItalic** создается и добавляется в семейство **TableDef** объекта **TableDef**. Это свойство определено в Microsoft Access, а не в ядре баз данных Microsoft Jet. Однако свойство применяется к объектам DAO. Поэтому, чтобы установить его из программы на Visual Basic, необходимо создать объект **Property**, соответствующий этому свойству, и добавить его в семейство **Properties** объекта DAO. Объект необходимо создавать только при первой установке значения свойства.

```
Sub CheckInherited()
   Dim dbs As Database, tdf As TableDef, qdf As QueryDef
   Dim strSQL As String

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Возвращает ссылку на таблицу «Заказы».
   Set tdf = dbs.TableDefs!Заказы
   ' Вызывает функцию SetAccessProperty.
   If SetAccessProperty(tdf, "DatasheetFontItalic", _
         dbBoolean, True) = True Then
      ' Создает объект QueryDef на основе таблицы «Заказы».
      strSQL = "SELECT * FROM Заказы WHERE СтранаПолучателя = 'США';"
      Set qdf = dbs.CreateQueryDef("ЗаказыСША", strSQL)
      ' Возвращает объект Property, указывающий на свойство.
      Debug.Print "Значение свойства Inherited, объект TableDef:", _
         tdf.Properties!DatasheetFontItalic.Inherited
      Debug.Print "Значение свойства Inherited, объект QueryDef:", _
         qdf.Properties!DatasheetFontItalic.Inherited
         ExitCheckInherited
   Else
      MsgBox "Значение свойства успешно не задано."
      ExitCheckInherited
   End If

ExitCheckInherited:
   Set dbs = Nothing
   Exit Sub
End Sub
```

Эта процедура вызывает следующую функцию, присваивающую значение свойству, создавая его при необходимости в семействе **Properties**:

```
Function SetAccessProperty(obj As Object, strName As String, _
      intType As Integer, varSetting As Variant) As Boolean
   Dim prp As Property
   Const conPropNotFound As Integer = 3270

   On Error GoTo ErrorSetAccessProperty
   ' Явная ссылка на семейство Properties.
   obj.Properties(strName) = varSetting
   obj.Properties.Refresh
   SetAccessProperty = True
```

```
ExitSetAccessProperty:
   Exit Function

ErrorSetAccessProperty:
   If Err = conPropNotFound Then
      ' Создает свойство, задает тип и начальное значение.
      Set prp = obj.CreateProperty(strName, intType, varSetting)
      ' Добавляет объект Property в семейство Properties.
      obj.Properties.Append prp
      obj.Properties.Refresh
      SetAccessProperty = True
      Resume ExitSetAccessProperty
   Else
      MsgBox Err & ": " & vbCrLf & Err.Description
      SetAccessProperty = False
      Resume ExitSetAccessProperty
   End If
End Function
```

Свойство LastModified (Microsoft Access), пример

В следующем примере определяется запись набора, которая была изменена позднее всего, и для нее устанавливается закладка:

```
Sub LastChangedRecord()
   Dim dbs As Database, rst As Recordset, varBook As Variant

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Создает динамический объект Recordset.
   Set rst = dbs.OpenRecordset("SELECT * FROM Заказы " & "ORDER BY
СтранаПолучателя;")
   With rst
      .FindFirst "СтранаПолучателя = 'Великобритания'"
      ' Изменяет запись и сохраняет изменения.
      .Edit
      !СтранаПолучателя = "США"
      .Update
   End With
   ' Проверяет, поддерживает ли набор записей закладки.
   If rst.Bookmarkable Then
      ' Получает последнюю измененную запись.
      varBook = rst.LastModified
      ' Устанавливает закладку на последнюю измененную запись.
      rst.Bookmark = varBook
   End If
   rst.Close
   Set dbs = Nothing
End Sub
```

Свойство LockEdits (Microsoft Access), пример

В следующем примере открывается динамический объект **Recordset** и его свойству **LockEdits** присваивается значение **False** (0). Это делает возможным нежесткую блокировку, при которой другие пользователи могут в любое время изменять записи базы данных. Если данные были изменены до вызова метода **Update**, Microsoft Access генерирует перехватываемую ошибку.

```
Sub ShowLockStatus()
   Dim dbs As Database, rst As Recordset

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Открывает динамический объект Recordset.
   Set rst = dbs.OpenRecordset("Клиенты", dbOpenDynaset)
   rst.LockEdits = False
   On Error Goto ErrorLockEdits
   ' Разрешает изменение.
   rst.Edit
   .                ' Изменяет записи.
   .
   .
   rst.Update       ' Попытка занесения изменений.

ExitLockEdits:
   rst.Close
   Set dbs = Nothing
   Exit Sub

ErrorLockEdits:
      If Err.Number = 3197 Then
         MsgBox "Данные были изменены другим пользователем."
         Resume Next
      Else
         MsgBox "Какая-то другая ошибка."
         Resume ExitLockEdits
      End If
End Sub
```

Свойство LoginTimeout (Microsoft Access), пример

В следующем примере свойству **LoginTimeout** присваивается значение 120 секунд, затем создается запрос и запускается в базе данных на сервере ODBC:

```
Sub Login()
   Dim dbs As Database
   Dim qdf As QueryDef, rst As Recordset

   ' Устанавливает время задержки равным 120 секундам.
   DBEngine.LoginTimeout = 120
   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Создает новый объект QueryDef.
   Set qdf = dbs.CreateQueryDef("Все сотрудники", "SELECT * FROM
Сотрудники;")
   qdf.Connect = "ODBC;DSN=Персонал; " _
      & "Database=HRSRVR; UID=Иванов; PWD=Сезам"
   ' Подключиться к серверу и запустить запрос.
   Set rst = qdf.OpenRecordset()
   ' Выполняет операции с набором записей.
   .
   .
   .
   rst.Close
   Set dbs = Nothing
End Sub
```

Свойство «Имя» (Name) - (Microsoft Access), пример

В следующем примере создаются два новых объекта **TableDef**, а затем им присваиваются имена. Имя первого объекта **TableDef** передается в качестве аргумента в метод **CreateTableDef**. Имя второго задается с помощью свойства **Имя (Name)** уже после создания объекта **TableDef**.

Следует отметить, что перед добавлением объекта **TableDef** в семейство **TableDefs** в таблице должны быть определены поля.

```
Sub NameNewTables()
   Dim dbs As Database
   Dim tdfDefinitions As TableDef, tdfSynonyms As TableDef

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Создает новый объект TableDef.
   Set tdfDefinitions = dbs.CreateTableDef("Описания")
   ' Создает второй объект TableDef.
   Set tdfSynonyms = dbs.CreateTableDef("")
   ' Задает значение свойства Имя (Name) второго объекта TableDef.
   tdfSynonyms.Name = "Синонимы"
   .                    ' Создает поля.
   .
   .
   dbs.TableDefs.Append tdfDefinitions
   dbs.TableDefs.Append tdfSynonyms
   Set dbs = Nothing
End Sub
```

Свойство NoMatch (Microsoft Access), пример

В следующем примере свойство **NoMatch** используется для проверки, успешно ли завершилось выполнение метода **FindFirst**:

```
Function ПоискСтраны() As Integer
   Dim dbs As Database, rst As Recordset
   Dim strCountry As String

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Создает динамический объект Recordset.
   Set rst = dbs.OpenRecordset("Заказы", dbOpenDynaset)
   strCountry = InputBox("Введите название страны.")
   rst.FindFirst "СтранаПолучателя = '" & strCountry & "'"
   If rst.NoMatch Then
      FindCountry = False
   Else
      FindCountry = True
      Debug.Print rst!КодЗаказа
   End If
   rst.Close
   Set dbs = Nothing
End Function
```

Свойство «Время ожидания ODBC» (ODBCTimeout) - (Microsoft Access), пример

В следующем примере свойству **Время ожидания (ODBCTimeout)** присваивается значение 120 секунд, затем создается запрос и запускается в базе данных на сервере ODBC:

```
Sub SetTimeout()
   Dim dbs As Database
   Dim qdf As QueryDef, rst As Recordset

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Создает новый объект QueryDef.
   Set qdf = dbs.CreateQueryDef("Все клиенты", _
      "SELECT * FROM Клиенты;")
   qdf.Connect = "ODBC;DSN=Персонал;SERVER=HRSRVR: " _
      & "UID=Иванов; PWD=Сезам"
   ' Подключается к серверу и запускает запрос.
   qdf.ODBCTimeout = 120
   Set rst = qdf.OpenRecordset()
   ' Выполняет операции с набором записей.
   .
   .
   .
   rst.Close
   Set dbs = Nothing
End Sub
```

Свойство OrdinalPosition (Microsoft Access), пример

В следующем примере изменяется значение свойства **OrdinalPosition** первого поля таблицы «Товары». Если посмотреть на таблицу в режиме конструктора до и после запуска этого примера, можно заметить, что поле «КодТовара» переместилось с первого места на последнее.

```
Sub SetPosition()
   Dim dbs As Database, tdf As TableDef
   Dim fldFirst As Field, fld As Field

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Возвращает ссылку на таблицу «Товары».
   Set tdf = dbs.TableDefs!Товары
   ' Возвращает ссылку на первое поле таблицы.
   Set fldFirst = tdf.Fields(0)
   ' Присваивает свойству OrdinalPosition значение последней позиции в
семействе.
   fldFirst.OrdinalPosition = tdf.Fields.Count
   ' Обновляет семейство Fields.
   tdf.Fields.Refresh
   ' Перебирает все поля и выводит их порядковые номера.
   For Each fld In tdf.Fields
      Debug.Print fld.Name, fld.OrdinalPosition
   Next fld
   Set dbs = Nothing
End Sub
```

Свойство PercentPosition (Microsoft Access), пример

В следующем примере текущая позиция записи в наборе записей выражается в процентах и выводится в окне сообщений:

```
Sub PercentOfRecords()
   Dim dbs As Database, rst As Recordset
   Dim strInput As String, strSQL As String

   strSQL = "SELECT * FROM Товары ORDER BY Марка;"
   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Открывает динамический объект Recordset.
   Set rst = dbs.OpenRecordset(strSQL, dbOpenDynaset)
   ' Переходит в конец набора записей.
   rst.MoveLast
   ' Возвращается в начало набора записей.
   rst.MoveFirst
   ' Предлагает пользователю ввести марку товара.
   strInput = InputBox("Пожалуйста введите полное название товара.")
   ' Находит первое вхождение.
   rst.FindFirst "Марка = '" & strInput & "'"
   ' Выводит текущее положение в процентах.
   MsgBox("Текущее положение в наборе записей равно " _
      & rst.PercentPosition & "%.")
   rst.Close
   Set dbs = Nothing
End Sub
```

Свойства Permissions и UserName (Microsoft Access), пример

В следующем примере путем создания объекта **User** и добавления его в семейство **Users** заданной по умолчанию рабочей области создается новая учетная запись пользователя. Затем новому пользователю дается разрешение на доступ ко всем таблицам в базе данных. Сначала в процедуре свойству **UserName** объекта **Container** с именем Tables присваивается имя нового пользователя, а затем свойству **Permissions** присваиваются нужные разрешения.

Следует отметить, что оператор **And** выполняет поразрядное сравнение для определения установленных атрибутов.

Примечание. При программировании систем доступа следует избегать включения пароля или разрешений в программу. Следующий пример служит лишь для демонстрационных целей.

```
Function SetPermissions(strName As String, strPID As String, _
      strPassword As String) As Boolean
   Dim dbs As Database, ctrTables As Container
   Dim wspDefault As Workspace, usrNew As User

   On Error GoTo ErrorSetPermissions
   ' Возвращает ссылку на заданную по умолчанию рабочую область.
   Set wspDefault = DBEngine.Workspaces(0)
   ' Создает объект User, задает имя, разрешения и пароль.
   Set usrNew = wspDefault.CreateUser(strName, strPID, strPassword)
   ' Добавляет объект в семейство Users заданной по умолчанию рабочей
области.
   wspDefault.Users.Append usrNew
   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Возвращает ссылку на семейство Tables.
   Set ctrTables = dbs.Containers!Tables
   ctrTables.UserName = usrNew.Name
   ' Устанавливает для пользователя разрешение на доступ к таблицам.
   ctrTables.Permissions = dbSecInsertData _
      And dbSecReplaceData And dbSecDeleteData
   SetPermissions = True

ExitSetPermissions
   Set dbs = Nothing
   Exit Function

ErrorSetPermissions
   MsgBox Err & ": " & Err.Description
   SetPermissions = False
   Resume ExitSetPermissions
End Function
```

В следующей процедуре пользователю предлагается ввести имя и пароль, после чего создается значение, отражающее разрешения:

```
Sub NewUserPrompt()
   Dim strName As String, strPassword As String
   Dim intR As Integer, strPID As String
   Dim blnReturn As Boolean

   ' Предлагает пользователю ввести имя и пароль.
   strName = InputBox("Введите ваше имя.")
   strPassword = InputBox("Введите пароль до 14 символов длиной.")
```

```
   ' Инициализирует генератор псевдослучайных чисел.
   Randomize
   ' Создает строку разрешений.
   For intR = 0 To 19
      strPID = strPID & Chr$(Int(256 * Rnd))
   Next intR
   ' Вызывает функцию SetPermissions.
   blnReturn = SetPermissions(strName, strPID, strPassword)
End Sub
```

Свойство Primary (Microsoft Access), пример

В следующем примере проверяется значение свойства **Primary** каждого индекса таблицы «Товары» и выводятся поля, являющиеся ключевыми:

```
Sub SetIndex()
   Dim dbs As Database, tdf As TableDef, fld As Field
   Dim idx As Index

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Возвращает ссылку на таблицу «Товары».
   Set tdf = dbs.TableDefs!Товары
   ' Перебирает элементы семейства Indexes объекта TableDef.
   For Each idx In tdf.Indexes
      ' Проверяет значение свойства Primary объекта Recordset.
      If idx.Primary Then
         For Each fld In idx.Fields
            Debug.Print fld.Name
         Next fld
      End If
   Next idx
   Set dbs = Nothing
End Sub
```

Свойство QueryTimeout (Microsoft Access), пример

В следующем примере задается значение свойства **QueryTimeout** текущей базы данных:

```
Sub SetTimeout()
   Dim dbs As Database

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   dbs.QueryTimeout = 120
   Set dbs = Nothing
End Sub
```

Свойство RecordCount (Microsoft Access), пример

В следующем примере на основе таблицы «Заказы» создается объект **Recordset** и определяется число записей в этом объекте:

```
Sub CountRecords()
   Dim dbs As Database, rst As Recordset

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Открывает табличный объект Recordset.
   Set rst = dbs.OpenRecordset("Заказы")
   Debug.Print rst.RecordCount
   rst.Close
   Set dbs = Nothing
End Sub
```

Свойство RecordsAffected (Microsoft Access), пример

В следующем примере выводится число записей, обновленных при выполнении запроса на изменение:

```
Sub RecordsUpdated()
   Dim dbs As Database, qdf As QueryDef
   Dim strSQL As String

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   strSQL = "UPDATE Сотрудники SET Должность = " _
      & "'Старший представитель' " & "WHERE Должность = 'Представитель';"
   ' Создает новый объект QueryDef.
   Set qdf = dbs.CreateQueryDef("ОбновитьДолжность", strSQL)
   ' Выполняет объект QueryDef.
   qdf.Execute
   Debug.Print qdf.RecordsAffected
   Set dbs = Nothing
End Sub
```

Свойство «Обязательное поле» (Required) - (Microsoft Access), пример

В следующем примере в таблице «Сотрудники» создается новый объект **Index** и задается значение его свойства **Обязательное поле (Required)**. Новый индекс состоит из двух полей: «Фамилия» и «Имя».

```
Sub NewIndex()
   Dim dbs As Database, tdf As TableDef, idx As Index
   Dim fldLastName As Field, fldFirstName As Field

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Возвращает ссылку на таблицу «Сотрудники».
   Set tdf = dbs.TableDefs!Сотрудники
   ' Создает новый индекс.
   Set idx = tdf.CreateIndex("ФИО")
   ' Создает и добавляет индексные поля.
   Set fldLastName = idx.CreateField("Фамилия", dbText)
   Set fldFirstName = idx.CreateField("Имя", dbText)
   idx.Fields.Append fldLastName
   idx.Fields.Append fldFirstName
   ' Обеспечить заполнение каждого поля в индексе.
   idx.Required = True
   ' Добавляет объект Index.
   tdf.Indexes.Append idx
   Set dbs = Nothing
End Sub
```

Свойства Size и Type (Microsoft Access), пример

В следующем примере создается новый объект **Field** и задаются значения его свойств **Size** и **Type**. Затем новый объект добавляется в семейство **Fields** таблицы «Сотрудники», находящейся в семействе **TableDefs** базы данных.

```
Sub NewField()
   Dim dbs As Database, tdf As TableDef
   Dim fld As Field

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Возвращает ссылку на таблицу «Сотрудники».
   Set tdf = dbs.TableDefs!Сотрудники
   ' Создает объект Field.
   Set fld = tdf.CreateField("ДнейВОтпуске")
   ' Задает значения свойств поля.
   fld.Type = dbText
   fld.Size = 20
   ' Добавляет поле в семейство Fields.
   tdf.Fields.Append fld
   Set dbs = Nothing
End Sub
```

Для установки значений свойств **Имя** (**Name)**, **Type** и **Size** можно также воспользоваться методом **CreateField**, передав ему в качестве аргументов значения *имя, тип* и *размер*.

```
Set fld = tdfEmployees.CreateField("ДнейВОтпуске", dbText, 20)
```

Свойство Sort (Microsoft Access), пример

В следующем примере свойство **Sort** используется для задания порядка сортировки динамического объекта **Recordset**, созданного на основе таблицы «Заказы». Записи переменной набора записей `rstSorted` будут отсортированы в алфавитном порядке по стране получателя.

Следует отметить, что сначала создается динамический объект **Recordset** и задается значение его свойства **Sort**, затем на его основе открывается второй динамический объект **Recordset**. Изменение значения свойства **Sort** первого объекта **Recordset** в действительности не влияет на содержащиеся в нем записи, поэтому, чтобы увидеть результат сортировки, необходимо создать второй объект **Recordset**.

```
Sub SortByCountry()
   Dim dbs As Database
   Dim rstOrders As Recordset, rstSorted As Recordset

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Создает динамический объект Recordset.
   Set rstOrders = dbs.OpenRecordset("Заказы", dbOpenDynaset)
   ' Задает порядок сортировки.
   rstOrders.Sort = "СтранаПолучателя"
   ' Создает второй динамический объект Recordset.
   Set rstSorted = rstOrders.OpenRecordset()
   ' Выполняет операции с набором записей.
   .
   .
   .
   rstOrders.Close
   rstSorted.Close
   Set dbs = Nothing
End Sub
```

Совет. В большинстве случает более эффективно создание второго объекта **Recordset** с заданными условиями за один шаг. Когда заранее известен нужный порядок сортировки, эффективнее использовать для создания объекта **Recordset** инструкцию SQL. В следующем примере показано, как, создав всего один набор записей, достичь того же результата, что и в предыдущем примере:

```
Sub CreateRecordsetWithSQL()
   Dim dbs As Database, rst As Recordset
   Dim strSelect As String

   Set dbs = CurrentDb
   strSelect = "SELECT * FROM Заказы ORDER BY СтранаПолучателя;"
   Set rst = dbs.OpenRecordset(strSelect)
   rst.MoveLast
   MsgBox "Набор записей содержит " & rst.RecordCount & " записей."
   rst.Close
End Sub
```

Свойства SourceField и SourceTable (Microsoft Access), пример

В следующем примере с помощью инструкции SQL, создающей псевдонимы для полей в двух различных таблицах базы данных, создается объект **Recordset**. Затем выводятся имена поля, исходной таблицы и исходного поля.

```
Sub SourceInfo()
   Dim dbs As Database, rst As Recordset, fld As Field
   Dim strSQL As String

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Создает инструкцию SQL.
   strSQL = "SELECT КодТовара As ШифрТовара, " _
      & "Категория As ТипТовара FROM Типы " _
      & "INNER JOIN Товары ON Типы.Тип " _
      & " = Товары.Тип;"
   ' Открывает динамический объект Recordset.
   Set rst = dbs.OpenRecordset(strSQL)
   For Each fld In rst.Fields
      ' Выводит имя поля.
      Debug.Print "Имя поля: "; fld.Name
      ' Выводит исходное имя таблицы.
      Debug.Print "Исходная таблица: "; fld.SourceTable
      ' Выводит исходное имя поля.
      Debug.Print "Исходное поле: "; fld.SourceField
      Debug.Print
   Next fld
   rst.Close
   Set dbs = Nothing
End Sub
```

Свойство SQL (Microsoft Access), пример

В следующем примере на основе таблицы «Заказы» создается запрос с параметрами. Запрос выбирает все заказы, дата размещения которых попадает в заданный пользователем диапазон.

```
Sub ДиапазонЗаказов()
   Dim dbs As Database, qdf As QueryDef, rst As Recordset

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Создает новый запрос.
   Set qdf = dbs.CreateQueryDef("ДиапазонЗаказов")
   ' Создает инструкцию SQL с параметрами.
   qdf.SQL = "PARAMETERS [ДатаС] DATETIME, [ДатаПо] DATETIME; " & _
      "SELECT * FROM Заказы WHERE ДатаРазмещения BETWEEN " _
      & "[ДатаС] AND [ДатаПо];"
   qdf.Parameters("ДатаС") = #1/1/96#
   qdf.Parameters("ДатаПо") = #1/31/96#
   ' Создает статический объект Recordset из объекта QueryDef.
   Set rst = qdf.OpenRecordset(dbOpenSnapshot)
   ' Выполняет операции с набором записей.
   .
   .
   .
   rst.Close
   Set dbs = Nothing
End Sub
```

Свойство Updatable (Microsoft Access), пример

В следующем примере в объект **Recordset** добавляется запись, если значение свойства **Updatable** этого объекта равно **True** (–1).

```
Sub UpdateData()
   Dim dbs As Database, rstUnknown As Recordset

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Открывает табличный объект Recordset.
   Set rstUnknown = dbs.OpenRecordset("Неизвестная Таблица ")
   ' Перед добавлением новой записи проверяет значение свойства Updatable.
   If rstUnknown.Updatable = True Then
      With rstUnknown
         .AddNew
         !НекотороеПоле = "Некоторые новые сведения"
         .Update
      End With
   End If
   rstUnknown.Close
   Set dbs = Nothing
End Sub
```

Свойства «Условие на значение» (ValidationRule) и «Сообщение об ошибке» (ValidationText) - (Microsoft Access), пример

В следующем примере задаются значения свойств **Условие на значение (ValidationRule)** и **Сообщение об ошибке (ValidationText)** для двух полей в таблице «Заказано». После установки значений этих свойств они могут быть просмотрены в режиме конструктора таблицы.

```
Sub SetValidation()
   Dim dbs As Database, tdf As TableDef
   Dim fldQuantity As Field, fldDiscount As Field

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Возвращает ссылку на таблицу «Заказано».
   Set tdf = dbs.TableDefs![Заказано]
   Set fldQuantity = tdf.Fields!Количество
   Set fldDiscount = tdf.Fields!Скидка
   ' Задает значения свойств «Условие на значение» (ValidationRule) и
«Сообщение об ошибке» (ValidationText).
   fldQuantity.ValidationRule = ">= 4"
   fldQuantity.ValidationText = "Значение в поле 'Количество' должно быть
не меньше четырех."
   fldDiscount.ValidationRule = "Between .05 and .30"
   fldDiscount.ValidationText = "Скидка должна попадать в диапазон от 5% до
30%."
   Set dbs = Nothing
End Sub
```

Инструкция SELECT (Microsoft Access)

В Microsoft Access имена полей (*поле1, поле2*), указанные в режиме SQL окна запроса, используются в режиме таблицы в качестве заголовков столбцов. Для того чтобы вывести в режиме таблицы другие заголовки столбцов, следует включать в инструкцию SQL зарезервированное слово AS. Зарезервированное слово AS позволяет вывести данные в режиме таблицы с заголовками столбцов, задающимися аргументами *псевдоним1, псевдоним2* и т.д. Использование предложения AS для указания псевдонима таблицы эквивалентно определению свойства **Псевдоним (Alias)** в окне свойств списка полей в режиме конструктора запроса.

Следующая конструкция задает заголовок столбца «Юбилей» для результирующей таблицы:

```
SELECT [ДатаРождения] AS Юбилей FROM Сотрудники;
```

Указание имен столбцов с помощью предложения AS становится необходимым при использовании статистических функций или в запросах, возвращающих слишком сложные или повторяющиеся имена полей. В следующем примере в режиме таблицы выводится результат подсчета числа сотрудников:

```
SELECT COUNT([КодСотрудника])
AS [Учет поголовья] FROM Сотрудники;
```

При работе с объектами доступа к данным (DAO) в программе Visual Basic аргументы *поле1, поле2* задают имена объектов **Field** (Поле) возвращаемого запросом объекта **Recordset** (Набор записей). При включении в инструкцию ключевого слова AS аргументы *псевдоним1, псевдоним2* определят заголовки столбцов, возвращаемые как имена объектов **Field** в результирующем объекте **Recordset**.

Предложение FROM (Microsoft Access)

При включении имени таблицы или запроса в предложение FROM инструкции SQL эта таблица или запрос автоматически добавляется в окно запроса.

Любая таблица или запрос, добавленные в окне запроса, автоматически включатся в предложение FROM соответствующей инструкции SQL.

Предложение GROUP BY (Microsoft Access)

Включение предложения GROUP BY в инструкцию SQL эквивалентно созданию итогового запроса в окне запроса и выбору для соответствующего поля операции «Группировка» в ячейке строки «Групповая операция».

Предложение WHERE (Microsoft Access)

В Microsoft Access указание условий отбора в предложении WHERE в режиме SQL эквивалентно вводу соответствующих условий в ячейку строки «Условие отбора» в бланке запроса по образцу. Если условие отбора создается в бланке запроса, то для просмотра соответствующего предложения WHERE в инструкции SQL можно переключиться в режим SQL.

Если предложение WHERE в инструкции SQL создается в режиме SQL, то после переключения в режим конструктора соответствующее условие будет введено в бланк запроса по образцу. Это относится ко всем запросам кроме запросов на объединение, просмотр которых возможен только в режиме SQL.

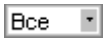
Предложение ORDER BY (Microsoft Access)

В Microsoft Access предложение ORDER BY задает сортировку данных по указанному полю или полям по возрастанию или по убыванию. Использование предложения ORDER BY эквивалентно выбору «по возрастанию» или «по убыванию» в ячейке строки «Сортировка» в бланке запроса по образцу.

Предикаты ALL, DISTINCT, DISTINCTROW, TOP (Microsoft Access)

В Microsoft Access использование зарезервированного слова DISTINCT эквивалентно выбору в режиме конструктора запроса значения «Да» для свойства **Уникальные значения (UniqueValues)** в окне свойств запроса.

Использование зарезервированного слова DISTINCTROW эквивалентно выбору в режиме конструктора запроса значения «Да» (значение по умолчанию) для свойства **Уникальные записи (UniqueRecords)** в окне свойств запроса.

Использование зарезервированного слова TOP эквивалентно указанию значения свойства **Набор значений (TopValues)** в окне свойств запроса в режиме конструктора запроса или вводу значения в поле **Набор значений** 

на панели инструментов «Конструктор запросов».

Использование зарезервированного слова PERCENT эквивалентно указанию символа процентов (%) в значении свойства **Набор значений (TopValues)** в окне свойств запроса или в поле **Набор значений**.

Инструкция DELETE (Microsoft Access)

В Microsoft Access выполнение инструкции DELETE не приводит к выводу результатов отчета в режиме таблицы. Для того чтобы проверить, какие записи будут удалены, следует предварительно выполнить запрос на выборку, в котором используются те же условия отбора записей, что и в запросе на удаление, просмотреть результаты в режиме таблицы и только после этого выполнить запрос на удаление записей.

Инструкция INSERT INTO (Microsoft Access)

Если создать в режиме SQL запрос на добавление с помощью инструкции INSERT INTO...VALUES, сохранить и закрыть его, а затем открыть снова, то предложение VALUES будет преобразовано в предложение SELECT. Это не повлияет на результат выполнения запроса.

Использование инструкции INSERT INTO эквивалентно определению свойства **Таблица-получатель (DestinationTable)** в окне свойств запроса на добавление в режиме конструктора запроса.

Описание PARAMETERS (Microsoft Access)

При выполнении запроса с параметрами Microsoft Access выводит приглашение пользователю ввести параметры, определяющие условия отбора в запросе. Это позволяет создать запрос, возвращающий записи на основе заданного пользователем условия.

В аргументах *текст*, перечисленных в описании PARAMETERS, задается текст, выводящийся в каждом из диалоговых окон, которые открываются для ввода соответствующего параметра при запуске запроса. Эти диалоговые окна создаются автоматически.

Использование описания PARAMETERS в режиме SQL эквивалентно описанию параметров в ячейке строки «Условие отбора» в бланке запроса.

Предложение PROCEDURE (Microsoft Access)

Если предложение PROCEDURE копируется в окно режима SQL, то Microsoft Access удалит это предложение из инструкции SQL после перехода в другой режим. Удаление предложения PROCEDURE не влияет на результат выполнения запроса. Если предложение PROCEDURE содержало описание параметров, Microsoft Access заменит зарезервированное слово PROCEDURE и аргумент *имя* зарезервированным словом PARAMETERS. Описание параметров удалено не будет.

Операция UNION (Microsoft Access)

В Microsoft Access аргументами операции UNION (*запрос1, запрос2,...запросN*) могут служить инструкция SELECT, имя сохраненного запроса Microsoft Access или имя сохраненной таблицы Microsoft Access с предшествующим зарезервированным словом TABLE.

Просмотр запроса на объединение возможен только в режиме SQL, нельзя просматривать такой запрос в бланке запроса.

Инструкция UPDATE (Microsoft Access)

Инструкция UPDATE не возвращает записи, поэтому при ее выполнении результаты не выводятся в режиме таблицы.

Если требуется подтвердить или отменить каждое изменение, используйте вместо запроса на обновление команду **Заменить** из меню **Правка** в режимах формы или таблицы.

Описание WITH OWNERACCESS OPTION (Microsoft Access)

Описание WITH OWNERACCESS OPTION, включенное в инструкцию SQL, дает возможность пользователям, в обычных условиях не обладающим разрешением на базовые таблицы, возможность выполнять запрос и просматривать его результаты. При наличии этого описания пользователю предоставляются те же права, что и владельцу запроса.

Например, владелец запроса по таблице «Сотрудники» может включить описание WITH OWNERACCESS OPTION в запрос, возвращающий все поля таблицы, кроме тех, в которых сохранятся адреса сотрудников. Пользователь, не имеющий разрешения на чтение таблицы «Сотрудники», будет иметь возможность выполнить запрос и просмотреть все поля, включенные в запрос. В результате, обеспечивается ограниченный доступ к данным в таблице.

Включение данного описания эквивалентно заданию в свойстве **Для запуска необходимы права (RunPermissions)** значения «Владельца» в окне свойств запроса в режиме конструктора запроса. Отсутствие описания WITH OWNERACCESS OPTION эквивалентно значению «Пользователя», задающемуся для данного свойства по умолчанию.

Статистические функции SQL (Microsoft Access)

Статистические функции SQL аналогичны статистическим функциям по подмножеству, однако, используются в другом контексте. В Microsoft Access допускается использование статистических функций SQL в инструкции SQL, создаваемой в режиме SQL окна запроса, или в инструкции SQL, входящей в программу Visual Basic. В отличие от этого, статистические функции по подмножеству допускают прямой вызов из программы Visual Basic.

В выражениях, определенных в бланке запроса, можно использовать как статистические функции SQL, так и статистические функции по подмножеству. Статистические функции SQL обычно используют в итоговых запросах и в перекрестных запросах.

В вычисляемых элементах управления в формах и отчетах используют оба типа статистических функций.

Функция Avg (Microsoft Access)

В Microsoft Access функцию **Avg** используют в бланке запроса, в инструкции SQL в режиме SQL окна запроса или в инструкции SQL в программе Visual Basic. Кроме того, функцию **Avg** используют для определения вычисляемого элемента управления в форме или отчете.

Функция **Avg** обычно применяется в итоговых запросах и в перекрестных запросах. Она дает одинаковые результаты при создании запроса в бланке запроса или в виде инструкции SQL в режиме SQL.

В бланке запроса допускается создание нового итогового запроса с помощью кнопки **Групповые операции**  на панели инструментов **Конструктор запросов**. При нажатии этой кнопки в бланк запроса добавляется строка **Групповая операция**. В ячейке этой строки выбирается статистическая функция, с помощью которой обрабатываются данные в соответствующем поле.

Предположим, например, что имеется таблица «Заказы», содержащая поля «СтоимостьДоставки» и «ГородПолучателя». Можно создать запрос, в котором будет выводиться средняя стоимость доставки заказов, отправленных в каждый город, название которого присутствует в таблице. Создайте новый итоговый запрос и переместите с помощью мыши поле «ГородПолучателя» в бланк запроса по образцу. В ячейке «Групповая операция» в столбце поля «ГородПолучателя» следует выбрать «Группировка». Переместите с помощью мыши поле «СтоимостьДоставки» в бланк запроса по образцу и выберите «Avg» в ячейке «Групповая операция» в столбце этого поля. При выполнении запроса будет выведена средняя стоимость доставки заказов для каждого города.

Для просмотра инструкции SQL, соответствующей данному запросу, следует переключиться в режим SQL. В данном примере Microsoft Access создает следующую инструкцию SQL:

```
SELECT ГородПолучателя, Avg(СтоимостьДоставки) AS СредняяЦенаДоставки FROM
Заказы GROUP BY ГородПолучателя;
```

Инструкцию SQL можно также использовать в программе Visual Basic. Например, в следующей программе создается динамический объект **Recordset** (набор записей) на основании описанной выше инструкции SQL:

```
Sub СредняяСтоимость()
   Dim dbs As Database, rst As Recordset, strSQL As String
   Set dbs = CurrentDb
   strSQL = "SELECT ГородПолучателя, Avg(СтоимостьДоставки) AS
СредняяЦенаДоставки " _
      & "FROM Заказы GROUP BY ГородПолучателя;"
   Set rst = dbs.OpenRecordset(strSQL)
   rst.MoveLast
   Debug.Print rst.RecordCount
   Set dbs = Nothing
End Sub
```

При использовании функции **Avg** для определения вычисляемого элемента управления следует указать в свойстве элемента управления **Данные (ControlSource)** выражение, включающее функцию **Avg**. Например, для того чтобы вывести в поле среднюю стоимость доставки для набора заказов, введите в ячейку свойства поля **Данные (ControlSource)** следующее выражение:

```
= Avg([СтоимостьДоставки])
```

Если функция **Avg** используется для определения вычисляемого элемента управления, можно ограничить набор записей, к которым применяется функция, с помощью свойства формы **Фильтр (Filter)**.

Функция Count (Microsoft Access)

В Microsoft Access функцию **Count** используют в бланке запроса, в инструкции SQL в режиме SQL окна запроса или в инструкции SQL в программе Visual Basic. Кроме того, функцию **Count** используют для определения вычисляемого элемента управления в форме или отчете.

Функция **Count** обычно применяется в итоговых запросах и в перекрестных запросах. Она дает одинаковые результаты при создании запроса в бланке запроса или в виде инструкции SQL в режиме SQL.

В бланке запроса допускается создание нового итогового запроса с помощью кнопки **Групповые операции**  на панели инструментов **Конструктор запросов**. При нажатии этой кнопки в бланк запроса добавляется строка **Групповая операция**. В ячейке этой строки выбирается статистическая функция, с помощью которой обрабатываются данные в соответствующем поле.

Самым быстрым способом подсчета всех записей в запросе является использование функции **Count(*)**. Функция **Count(*)** применяется также в запросе в вычисляемом поле.
Предположим, например, что имеется таблица «Заказы», содержащая поля «КодЗаказа» и «ГородПолучателя». Можно создать запрос, в котором будет выводиться число заказов, отправленных в каждый город. Создайте новый итоговый запрос и переместите с помощью мыши поле «ГородПолучателя» в бланк запроса по образцу. В ячейке строки «Групповая операция» под полем «ГородПолучателя» выберите «Группировка».

Далее, создайте вычисляемое поле с помощью следующего выражения, введенного в новую ячейку в строке «Поле»:

```
ЧислоЗаказов: Count(*)
```

В строке **Групповая операция** в этом столбце выберите «Выражение». При выполнении запроса будет выведено число заказов, отправленных в каждый город.

Для просмотра инструкции SQL, соответствующей данному запросу, следует переключиться в режим SQL. В данном примере Microsoft Access создает следующую инструкцию SQL:

```
SELECT ГородПолучателя, Count(*) AS ЧислоЗаказов
FROM Заказы GROUP BY ГородПолучателя;
```

Тот же результат можно получить, если перенести с помощью мыши поле «КодЗаказа» в бланк запроса по образцу и выбрать **Count** в строке **Групповая операция**. Такой запрос будет выполняться несколько медленнее, чем запрос с использованием функции **Count(*)**. Обратите внимание на отличия следующей инструкции SQL от предыдущей:

```
SELECT ГородПолучателя, Count(КодЗаказа) AS ЧислоЗаказов FROM Заказы GROUP
BY ГородПолучателя;
```

Инструкцию SQL можно также использовать в программе Visual Basic. Например, в следующей программе создается динамический объект **Recordset** (набор записей) с помощью более быстрой из приведенных выше инструкций SQL:

```
Sub ПодсчетЗаказов()
   Dim dbs As Database, rst As Recordset, strSQL As String
   Set dbs = CurrentDb
   strSQL = "SELECT ГородПолучателя, Count(*) AS ЧислоЗаказов " _
      & "FROM Заказы GROUP BY ГородПолучателя;"
   Set rst = dbs.OpenRecordset(strSQL)
   rst.MoveLast
   Debug.Print rst.RecordCount
   Set dbs = Nothing
End Sub
```

При использовании функции **Count** для определения вычисляемого элемента управления следует указать в свойстве элемента управления **Данные (ControlSource)** выражение, включающее функцию **Count**. Например, для того чтобы вывести в поле число заказов, введите в ячейку свойства поля **Данные (ControlSource)** следующее выражение.

```
=Count([КодЗаказа])
```

Если функция **Count** используется для определения вычисляемого элемента управления, можно ограничить набор записей, к которым применяется функция, с помощью свойства формы **Фильтр (Filter)**.

Функции First, Last (Microsoft Access)

В Microsoft Access функции **First** и **Last** используют в бланке запроса, в инструкции SQL в режиме SQL окна запроса или в инструкции SQL в программе Visual Basic. Кроме того, функции **First** и **Last** используют для определения вычисляемого элемента управления в форме или отчете.

Функции **First** и **Last** особенно полезны в вычисляемых элементах управления в отчете. Например, если имеется отчет «Заказы» с группировкой по полю «СтранаПолучателя» и с сортировкой по полю «ДатаРазмещения», функции **First** и **Last** позволят вывести в вычисляемых полях даты самого раннего и самого позднего заказа в каждой группе. Для того чтобы выполнить группировку по полю «СтранаПолучателя», следует нажать кнопку **Сортировка и группировка**  на панели инструментов **конструктор отчета**. Выберите поле «СтранаПолучателя» в ячейке столбца «Поле/выражение» и задайте значение «Да» для свойств **Заголовок группы (GroupHeader)** и **Примечание группы (GroupFooter)**. В режиме конструктора создайте два новых поля в области примечаний группы «СтранаПолучателя» и определите свойство **Данные (ControlSource)** для каждого их полей с помощью следующих выражений:

```
=First([ДатаРазмещения])
=Last([ДатаРазмещения])
```

При переходе в режим предварительного просмотра в этих полях для каждой группы будут выведены даты самого раннего и самого позднего заказа.

Примечание. Если требуется вернуть первую или последнюю запись из набора записей в Microsoft Access 97, следует создать запрос с сортировкой по возрастанию или по убыванию и задать для свойства **Набор значений (TopValues)** значение 1. Более подробное описание см. в разделе справки для свойства **Набор значений (TopValues)**. В Visual Basic можно также создать в запросе отсортированный результирующий набор записей и использовать метод объектов доступа к данным **MoveFirst** или **MoveLast** для возвращения первой или последней записи.

Функции Min, Max (Microsoft Access)

В Microsoft Access функции **Min** и **Max** используют в бланке запроса, в инструкции SQL в режиме SQL окна запроса или в инструкции SQL в программе Visual Basic. Кроме того, функции **Min** и **Max** используют для определения вычисляемого элемента управления в форме или отчете.

Функции **Min** и **Max** особенно полезны в итоговых запросах и в перекрестных запросах. Их применение дает одинаковые результаты при создании запроса в бланке запроса или в виде инструкции SQL в режиме SQL.

В бланке запроса допускается создание нового итогового запроса с помощью кнопки **Групповые операции**  на панели инструментов **Конструктор запросов**. При нажатии этой кнопки в бланк запроса добавляется строка **Групповая операция**. В ячейке этой строки выбирается статистическая функция, с помощью которой обрабатываются данные в соответствующем поле.

Предположим, например, что имеется таблица «Заказы», содержащая поля «СтоимостьДоставки» и «ГородПолучателя». Можно создать запрос, в котором будет выводиться минимальная стоимость доставки заказов для каждого города. Создайте новый итоговый запрос и переместите с помощью мыши поле «ГородПолучателя» в бланк запроса по образцу. В ячейке «Групповая операция» в столбце поля «ГородПолучателя» следует выбрать «Группировка». Переместите с помощью мыши поле «СтоимостьДоставки» в бланк запроса по образцу и выберите «Min» в ячейке «Групповая операция» в столбце этого поля. При выполнении запроса будет выведено минимальное значение стоимости доставки заказа для каждого города.

Для просмотра инструкции SQL, соответствующей данному запросу, следует переключиться в режим SQL. В данном примере Microsoft Access создает следующую инструкцию SQL:

```
SELECT ГородПолучателя, Min(СтоимостьДоставки) AS МинСтоимость
FROM Заказы GROUP BY ГородПолучателя;
```

Инструкцию SQL можно также использовать в программе Visual Basic. Например, в следующей программе создается динамический объект **Recordset** (набор записей) на основании описанной выше инструкции SQL:

```
Sub МинСтоимость()
   Dim dbs As Database, rst As Recordset, strSQL As String

   Set dbs = CurrentDb
   strSQL = "SELECT ГородПолучателя, Min(СтоимостьДоставки) AS МинСтоимость
" _
   & "FROM Заказы GROUP BY ГородПолучателя;"
   Set rst = dbs.OpenRecordset(strSQL)
   rst.MoveLast
   Debug.Print rst.RecordCount
   Set dbs = Nothing
End Sub
```

При использовании функций **Min** или **Max** для определения вычисляемого элемента управления следует указать в свойстве элемента управления **Данные (ControlSource)** выражение, содержащее одну из этих функций. Например, для того чтобы вывести в поле минимальное значение стоимости доставки, введите в ячейку свойства поля **Данные (ControlSource)** следующее выражение:

```
=Min([СтоимостьДоставки])
```

Если функции **Min** или **Max** используются для определения вычисляемого элемента управления, можно ограничить набор записей, к которым применяется функция, с помощью свойства формы **Фильтр (Filter)**.

Функции StDev, StDevP (Microsoft Access)

В Microsoft Access функции **StDev** и **StDevP** используют бланке запроса, в инструкции SQL в режиме SQL окна запроса или в инструкции SQL в программе Visual Basic. Кроме того, функции **StDev** и **StDevP** используют для определения вычисляемого элемента управления в форме или отчете.

Функции **StDev** и **StDevP** особенно полезны в итоговых запросах и в перекрестных запросах. Их применение дает одинаковые результаты при создании запроса в бланке запроса или в виде инструкции SQL в режиме SQL.

В бланке запроса допускается создание нового итогового запроса с помощью кнопки **Групповые операции**  на панели инструментов **Конструктор запросов**. При нажатии этой кнопки в бланк запроса добавляется строка **Групповая операция**. В ячейке этой строки выбирается статистическая функция, с помощью которой обрабатываются данные в соответствующем поле.

Предположим, например, что имеется таблица «Заказы», содержащая поля «СтоимостьДоставки» и «ГородПолучателя». Можно создать запрос, в котором будет выводиться несмещенное отклонение для выборки значений стоимости доставки заказов в каждый город. Создайте новый итоговый запрос и переместите с помощью мыши поле «ГородПолучателя» в бланк запроса по образцу. В ячейке «Групповая операция» в столбце поля «ГородПолучателя» следует выбрать «Группировка». Переместите с помощью мыши поле «СтоимостьДоставки» в бланк запроса по образцу и выберите «StDev» в ячейке «Групповая операция» в столбце этого поля. При выполнении запроса будет выведено значение несмещенного отклонение для стоимости доставки заказов в каждый город.

Для просмотра инструкции SQL, соответствующей данному запросу, следует переключиться в режим SQL. В данном примере Microsoft Access создает следующую инструкцию SQL:

```
SELECT ГородПолучателя, StDev(СтоимостьДоставки) AS СтОтклонение
FROM Заказы GROUP BY ГородПолучателя;
```

Инструкцию SQL можно также использовать в программе Visual Basic. Например, в следующей программе создается динамический объект **Recordset** (набор записей) на основании описанной выше инструкции SQL:

```
Sub ОтклСтоимости()
   Dim dbs As Database, rst As Recordset, strSQL As String

   Set dbs = CurrentDb
   strSQL = "SELECT ГородПолучателя, StDev(СтоимостьДоставки) " _
   & "AS СтОтклонение, ГородПолучателя FROM Заказы " _
   & "GROUP BY ГородПолучателя;"
   Set rst = dbs.OpenRecordset(strSQL)
   rst.MoveLast
   Debug.Print rst.RecordCount
   Set dbs = Nothing
End Sub
```

При использовании функций **StDev** и **StDevP** для определения вычисляемого элемента управления следует указать в свойстве элемента управления **Данные (ControlSource)** выражение, содержащее одну из этих функций. Например, для того чтобы вывести в поле значение несмещенного отклонение для стоимости доставки заказов, введите в ячейку свойства поля **Данные (ControlSource)** следующее выражение:

```
=StDev([СтоимостьДоставки])
```

Если функции **StDev** и **StDevP** используются для определения вычисляемого элемента управления, можно ограничить набор записей, к которым применяется функция, с помощью

свойства формы **Фильтр (Filter)**.

Функция Sum (Microsoft Access)

В Microsoft Access функцию **Sum** используют в бланке запроса, в инструкции SQL в режиме SQL окна запроса или в инструкции SQL в программе Visual Basic. Кроме того, функцию **Sum** используют для определения вычисляемого элемента управления в форме или отчете.

Функция **Sum** обычно применяется в итоговых запросах и в перекрестных запросах. Она дает одинаковые результаты при создании запроса в бланке запроса или в виде инструкции SQL в режиме SQL.

В бланке запроса допускается создание нового итогового запроса с помощью кнопки **Групповые операции**  на панели инструментов **Конструктор запросов**. При нажатии этой кнопки в бланк запроса добавляется строка **Групповая операция**. В ячейке этой строки выбирается статистическая функция, с помощью которой обрабатываются данные в соответствующем поле.

Предположим, например, что имеется таблица «Заказы», содержащая поля «СтоимостьДоставки» и «ГородПолучателя». Можно создать запрос, в котором будет выводиться суммарное значение стоимости доставки заказов для каждого города. Создайте новый итоговый запрос и переместите с помощью мыши поле «ГородПолучателя» в бланк запроса по образцу. В ячейке «Групповая операция» в столбце поля «ГородПолучателя» следует выбрать «Группировка». Переместите с помощью мыши поле «СтоимостьДоставки» в бланк запроса по образцу и выберите «Sum» в ячейке «Групповая операция» в столбце этого поля. При выполнении запроса будет выведена суммарное значение стоимости доставки заказов для каждого города.

Для просмотра инструкции SQL, соответствующей данному запросу, следует переключиться в режим SQL. В данном примере Microsoft Access создает следующую инструкцию SQL.

```
SELECT ГородПолучателя, Sum(СтоимостьДоставки) AS СуммаЗаказов
FROM Заказы GROUP BY ГородПолучателя;
```

Инструкцию SQL можно также использовать в программе Visual Basic. Например, в следующей программе создается динамический объект **Recordset** (набор записей) на основании описанной выше инструкции SQL:

```
Sub СумСтоимость()
   Dim dbs As Database, rst As Recordset, strSQL As String

   Set dbs = CurrentDb
   strSQL = "SELECT ГородПолучателя, Sum(СтоимостьДоставки) AS СуммаЗаказов
" _
   & "FROM Заказы GROUP BY ГородПолучателя;"
   Set rst = dbs.OpenRecordset(strSQL)
   rst.MoveLast
   Debug.Print rst.RecordCount
   Set dbs = Nothing
End Sub
```

При использовании функции **Sum** для определения вычисляемого элемента управления следует указать в свойстве элемента управления **Данные (ControlSource)** выражение, включающее функцию **Sum**. Например, для того чтобы вывести в поле сумму стоимости доставки заказов, введите в ячейку свойства поля **Данные (ControlSource)** следующее выражение:

```
=Sum([СтоимостьДоставки])
```

Если функция **Sum** используется для определения вычисляемого элемента управления, можно ограничить набор записей, к которым применяется функция, с помощью свойства формы **Фильтр (Filter)**.

Функции Var, VarP (Microsoft Access)

В Microsoft Access функции **Var** и **VarP** используют в бланке запроса, в инструкции SQL в режиме SQL окна запроса или в инструкции SQL в программе Visual Basic. Кроме того, функцию **Var** и **VarP** используют для определения вычисляемого элемента управления в форме или отчете.

Функции **Var** и **VarP** особенно полезны в итоговых запросах и в перекрестных запросах. Их применение дает одинаковые результаты при создании запроса в бланке запроса или в виде инструкции SQL в режиме SQL.

В бланке запроса допускается создание нового итогового запроса с помощью кнопки **Групповые операции**  на панели инструментов **Конструктор запросов**. При нажатии этой кнопки в бланк запроса добавляется строка **Групповая операция**. В ячейке этой строки выбирается статистическая функция, с помощью которой обрабатываются данные в соответствующем поле.

Предположим, например, что имеется таблица «Заказы», содержащая поля «СтоимостьДоставки» и «ГородПолучателя». Можно создать запрос, в котором будет выводиться значение несмещенной дисперсии для выборки значений стоимости доставки заказов в каждый город. Создайте новый итоговый запрос и переместите с помощью мыши поле «ГородПолучателя» в бланк запроса по образцу. В ячейке «Групповая операция» в столбце поля ГородПолучателя следует выбрать «Группировка». Переместите с помощью мыши поле «СтоимостьДоставки» в бланк запроса по образцу и выберите «Var» в ячейке «Групповая операция» в столбце этого поля. При выполнении запроса будет выведено значение дисперсии для стоимости доставки заказов в каждый город.

Для просмотра инструкции SQL, соответствующей данному запросу, следует переключиться в режим SQL. В данном примере Microsoft Access создает следующую инструкцию SQL:

```
SELECT ГородПолучателя, Var(СтоимостьДоставки) AS Дисперсия
FROM Заказы GROUP BY ГородПолучателя;
```

Инструкцию SQL можно также использовать в программе Visual Basic. Например, в следующей программе создается динамический объект **Recordset** (набор записей) на основании описанной выше инструкции SQL:

```
Sub ДиспСтоимости()
   Dim dbs As Database, rst As Recordset, strSQL As String

   Set dbs = CurrentDb
   strSQL = "SELECT ГородПолучателя, Var(СтоимостьДоставки) AS Дисперсия "
_
   & "FROM Заказы GROUP BY ГородПолучателя;"
   Set rst = dbs.OpenRecordset(strSQL)
   rst.MoveLast
   Debug.Print rst.RecordCount
   Set dbs = Nothing
End Sub
```

При использовании функций **Var** и **VarP** для определения вычисляемого элемента управления следует указать в свойстве элемента управления **Данные (ControlSource)** выражение, содержащее одну из этих функций. Например, для того чтобы вывести в поле значение дисперсии для стоимости доставки заказов, введите в ячейку свойства поля **Данные (ControlSource)** следующее выражение:

```
=Var([СтоимостьДоставки])
```

Если функции **Var** или **VarP** используются для определения вычисляемого элемента управления, можно ограничить набор записей, к которым применяется функция, с помощью свойства формы **Фильтр (Filter)**.

Оператор Between...And (Microsoft Access)

В Microsoft Access оператор **Between...And** используют в выражениях в запросе или в вычисляемом элементе управления в форме или отчете.

Оператор **Between...And** включают в ячейку строки «Условие» в бланке запроса при создании запроса с параметрами. Использование оператора **Between...And** позволяет вывести на экран приглашение пользователю ввести значения, задающие диапазон отбираемых в запросе данных. Выражение, определяющее аргумент *значение1*, задает приглашение ввести нижнюю границу диапазона, а выражение, определяющее аргумент *значение2*, задает приглашение ввести верхнюю границу.

Предположим, например, что имеется таблица «Заказы», содержащая поле «ДатаРазмещения». Создайте новый запрос и переместите с помощью мыши поле «ДатаРазмещения» в первую ячейку строки «Поле» в бланке запроса по образцу. В ячейку строки «Условие отбора» в том же столбце введите следующую инструкцию:

```
Between [Введите начальную дату:] And [Введите конечную дату:]
```

При запуске запроса вначале будет открыто диалоговое окно с приглашением «Введите начальную дату:». После ввода значения в это диалоговое окно будет открыто второе диалоговое окно с приглашением «Введите конечную дату:». Если пользователь ввел допустимые значения дат, в запросе будут отобраны записи со значениями в поле «ДатаРазмещения», совпадающие с введенными граничными значениями или лежащие между ними.

Оператор **Between...And** часто используют для проверки, попадает ли значение элемента управления в указанный диапазон числовых значений. В следующем примере проверяется, доставлен ли заказ в точку назначения, относящуюся к определенному диапазону почтовых кодов. Если значение поля «ПочтовыйИндекс» лежит в интервале от 198101 до 198199, функция **IIf** возвращает «Местный», иначе она возвращает «Не местный».

```
= IIf([ПочтовыйИндекс] Between "198101" And "198199", "Местный", "Не местный")
```

Если один из аргументов оператора **Between...And**, *выражение*, *значение1* или *значение2*, имеет значение **Null,** то оператор **Between...And** возвращает значение **Null**.

Оператор In (Microsoft Access)

В Microsoft Access оператор **In** используют в выражениях в запросе или в вычисляемом элементе управления в форме или отчете.

Оператор **In** включают в выражение в запросе при необходимости указать набор значений в условии отбора. Предположим, например, что имеется таблица «Заказы», содержащая поля «СтранаПолучателя» и «Заказ», и требуется создать запрос, в котором отбираются все заказы, отправленные в Латвию, Литву или Эстонию. Создайте в окне запроса новый запрос и добавьте в него таблицу «Заказы». Переместите с помощью мыши в бланк запроса по образцу поля «Заказ» и «СтранаПолучателя». В столбце поля «СтранаПолучателя» введите следующее выражение в ячейку строки "Условие отбора":

```
In('Латвия', 'Литва', 'Эстония')
```

При выполнении запроса будут выведены все заказы, отправленные в одну из этих стран.

Тот же результат можно получить с помощью следующего выражения в строке «Условие отбора:

```
»Латвия" Or "Литва" Or "Эстония"
```

Условия отбора, содержащие длинный список значений, удобнее записывать с помощью оператора **In**, а не с помощью оператора **Or**. Инструкции SQL, записанные с помощью оператора **In**, обычно оказываются короче.

Оператор **In** позволяет провести проверку, принадлежит ли значение элемента управления указанному списку значений. В следующем примере с помощью функции **IIf** проверяется, входит ли город получателя в список городов, для которых был выполнен заказ.

```
= IIf([ГородПолучателя] In ('Москва','Клин','Тверь'), "Выполнен", "Не
выполнен")
```

Оператор SQL Like (Microsoft Access)

В Microsoft Access оператор **Like** используют в выражениях в запросе или в вычисляемом элементе управления в форме или отчете.

Оператор **Like** позволяет задавать на бланке запроса условия неточного совпадения с образцом. Например, выражение **Like "Б*"** в ячейке строки «Условие отбора» задает отбор всех значений поля, начинающихся с буквы «Б».

В запросе с параметрами оператор **Like** задает вывод для пользователя приглашения ввести образец поиска. Предположим, что имеется таблица «Сотрудники», содержащая поле «Фамилия». В окне запроса создайте новый запрос, добавьте в него таблицу «Сотрудники» и переместите с помощью мыши поле «Фамилия» в бланк запроса. Введите следующее выражение в ячейку строки «Условие отбора»:

```
Like [Введите первые буквы фамилии:]&"*"
```

При запуске этого запроса будет открыто диалоговое окно с приглашением «Введите первые буквы фамилии:». Если пользователь введет буквы **Ку**, в запросе будет проведен поиск по образцу «Ку*», т.е. будут отобраны все фамилии, начинающиеся с букв «Ку».

Допускается использование оператора **Like** в выражениях, задающих свойство **Условие на значение (ValidationRule)** или условие в макросе. Так ограничивают данные, которые можно ввести в поле при неточном указании допустимых значений. Например, выражение

```
Like "Д[а-й]###"
```

введенное в ячейку свойства **Условие на значение (ValidationRule)**, позволяет вводить в поле строковое значение, начинающееся с буквы «Д», за которой следует одна из букв от «а» до «й» и три цифры.

Выражения SQL (Microsoft Access)

Допускается использование многих функций Visual Basic в строках SQL, определяющих инструкции SQL в программе Visual Basic, в режиме SQL окна запроса или в бланке запроса. Существует возможность написания собственных функций Visual Basic и их использование в строках SQL.

Включаемые в строки SQL функции должны возвращать значение, являющееся либо строкой, либо значением типа **Variant.** Также, любая операция в функции выполняется только один раз. Не допускается включение в строку функции, которая должна быть выполнена над каждой записью в наборе данных, поскольку инструкция SQL передается в ядро базы данных Microsoft Jet только один раз. Например, нельзя создать функцию, имеющую в качестве аргумента значение поля.

Можно включать в программу функции Visual Basic в инструкции SQL, с помощью которых определяется объект **QueryDef** (запрос) или объект **Recordset** типа динамического набора записей или статического набора записей. В Microsoft Access допускается использование функций Visual Basic в инструкции SQL, условиях отбора в бланке запроса, а также в выражениях, определяющих вычисляемые поля.

Выражения SQL (Microsoft Access), пример

В следующем примере с помощью инструкции SQL создается объект **Recordset** типа динамического набора записей. В предложение WHERE инструкции SQL включена функция **Year**, определяющая отбор заказов, размещенных в 1996 г.

```
Sub Orders96()
   Dim dbs As Database, rst As Recordset, strSQL As String
   Dim fld As Field

   Set dbs = CurrentDb
   strSQL = "SELECT DISTINCTROW Заказ, ДатаРазмещения " _
      & "FROM Заказы WHERE ((Year([ДатаРазмещения])=1996));"
   Set rst = dbs.OpenRecordset(strSQL, dbOpenDynaset)
   rst.MoveLast
   Debug.Print rst.RecordCount
End Sub
```

В следующем примере показано, как включать функцию Visual Basic в инструкцию SQL в режиме SQL окна запроса. Приведенная ниже инструкция SQL определяет запрос, в котором выводится поле «НазваниеПолучателя» из таблицы «Заказы», затем с помощью функции **Len** подсчитывается число символов, содержащихся в этом поле в каждой записи, а результат расчета выводится в отдельном элементе управления:

```
SELECT DISTINCTROW НазваниеПолучателя, Len([НазваниеПолучателя]) AS
ДлинаНазванияПолучателя
FROM Заказы;
```

Этот же запрос можно определить в бланке запроса. Создайте новый запрос и добавьте в него таблицу «Заказы». Переместите с помощью мыши поле «НазваниеПолучателя» в ячейку строки «Поле» в первом столбце бланка. В ячейку строки «Поле» в соседнем столбце введите следующее выражение, определяющее вычисляемое поле:

```
ДлинаНазванияПолучателя: Len([НазваниеПолучателя])
```

Функция Avg (Microsoft Access), пример

В данном примере предполагается**,** что имеется таблица «Заказы», содержащая поле «СтоимостьДоставки». Функция **Avg** позволяет найти среднюю стоимость доставки для заказов, в которых стоимость доставки превышает 100 000 рублей. Следующая инструкция вводится в окно запроса в режиме SQL:

```
SELECT Avg([СтоимостьДоставки]) AS [СредняяЦенаДоставки] FROM Заказы WHERE
[СтоимостьДоставки] > 100000;
```

В следующем примере создается вычисляемое поле, в котором выводится средняя стоимость доставки для всех записей в форме, базирующейся на той же таблице «Заказы». Откройте новую форму и задайте для свойства **Источник записей (RecordSource)** значение «Заказы». Задайте в свойстве поля **Данные (ControlSource)** следующее выражение. Если требуется ограничить отбираемый набор записей, например, отобрать заказы со стоимость выше 100 000 рублей, следует задать соответствующее значение свойства формы **Фильтр (Filter)**.

```
=Avg([СтоимостьДоставки])
```

Функция Count (Microsoft Access), пример

В данном примере предполагается, что имеется таблица «Заказы», содержащая поле «СтранаПолучателя». Функция **Count** позволяет найти число заказов, отправленных в Литву. Следующая инструкция вводится в окно запроса в режиме SQL:

```
SELECT Count([СтранаПолучателя]) AS [ДляЛитвы] FROM Заказы WHERE
[СтранаПолучателя] = 'Литва';
```

В следующем примере создается вычисляемое поле, в котором выводится число заказов из той же таблицы «Заказы». Откройте новую форму и задайте для свойства **Источник записей (RecordSource)** значение «Заказы». Задайте в свойстве поля **Данные (ControlSource)** следующее выражение. Если требуется ограничить отбираемый набор записей, например, отобрать заказы, отправленные в Литву, следует задать соответствующее значение свойства формы **Фильтр (Filter)**.

```
=Count([СтранаПолучателя])
```

Функции Min, Max (Microsoft Access), пример

В данном примере предполагается, что имеется таблица «Заказы», содержащая поля «СтоимостьДоставки» и «СтранаПолучателя». Функции **Min** и **Max** позволят найти минимальное и максимальное значения стоимости доставки заказов в Литву. Следующие инструкции вводятся в окно запроса в режиме SQL:

```
SELECT Min([СтоимостьДоставки]) AS [МинЦенаДоставки] FROM Заказы WHERE
[СтранаПолучателя] = 'Литва';

SELECT Max([СтоимостьДоставки]) AS [МаксЦенаДоставки] FROM Заказы WHERE
[СтранаПолучателя] = 'Литва';
```

В следующем примере создается вычисляемое поле, в котором выводится минимальное значение стоимости доставки для всех записей в той же таблице «Заказы». Откройте новую форму и задайте для свойства **Источник записей (RecordSource)** значение «Заказы». Задайте в свойстве поля **Данные (ControlSource)** следующее выражение. Если требуется ограничить отбираемый набор записей, например, провести расчет для заказов, отправленных в Литву, следует задать соответствующее значение свойства формы **Фильтр (Filter)**.

```
=Min([СтоимостьДоставки])
```

StDev, StDevP Functions Example (Microsoft Access)

В данном примере предполагается, что имеется таблица «Заказы», содержащая поля «СтоимостьДоставки» и «СтранаПолучателя». Функции **StDev** и **StDevP** позволят найти значения несмещенного и смещенного отклонения для выборки значений стоимости доставки заказов, отправленных в Литву. Следующие инструкции вводятся в окно запроса в режиме SQL:

```
SELECT StDev([СтоимостьДоставки]) AS [НесмещенноеОтклонение] FROM Заказы
WHERE [СтранаПолучателя] = 'Литва';

SELECT StDevP([СтоимостьДоставки]) AS [СмещенноеОтклонение] FROM Заказы
WHERE [СтранаПолучателя] = 'Литва';
```

В следующем примере создается вычисляемое поле, в котором выводится значение несмещенного отклонения стоимости доставки для всех записей в той же таблице «Заказы». Откройте новую форму и задайте для свойства **Источник записей (RecordSource)** значение «Заказы». Задайте в свойстве поля **Данные (ControlSource)** следующее выражение. Если требуется ограничить отбираемый набор записей, например, провести расчет для заказов, отправленных в Литву, следует задать соответствующее значение свойства формы **Фильтр (Filter)**.

```
=StDev([СтоимостьДоставки])
```

Функция Sum (Microsoft Access), пример

В данном примере предполагается, что имеется таблица «Заказы», содержащая поля «КодЗаказа» и «СтранаПолучателя», и таблица «Заказано», содержащая поля «Цена» и «Количество». Функция **Sum** позволяет найти общую стоимость заказов, отправленных в Литву. Следующая инструкция вводится в окно запроса в режиме SQL:

```
SELECT Sum([Цена]*[ Количество]) AS [ОбщийЗаказ]
FROM Заказы INNER JOIN [Заказано] ON Заказы.[КодЗаказа] = [Заказано].
[КодЗаказа]
WHERE ([СтранаПолучателя] = 'Литва');
```

В следующем примере создается вычисляемое поле, в котором выводится общая сумма заказов для всех записей в таблице «Заказано». Откройте новую форму и задайте для свойства **Источник записей (RecordSource)** значение «Заказано». Задайте в свойстве поля **Данные (ControlSource)** следующее выражение. Если требуется ограничить отбираемый набор записей, например, провести расчет для заказов, отправленных в Литву, следует задать соответствующее значение свойства формы **Фильтр (Filter)**.

```
=Sum([Цена]*[Количество])
```

Функции Var, VarP (Microsoft Access), примеры

В данном примере предполагается, что имеется таблица «Заказы», содержащая поле «СтоимостьДоставки». Функции **Var** и **VarP** позволят найти значения несмещенной и смещенной дисперсии для выборки заказов, отправленных в Литву. Следующие инструкции вводятся в окно запроса в режиме SQL:

```
SELECT Var([СтоимостьДоставки]) AS [НесмещеннаяДисперсия] FROM Заказы WHERE
[СтранаПолучателя] = 'Литва';

SELECT VarP([СтоимостьДоставки]) AS [СмещеннаяДисперсия] FROM Заказы WHERE
[СтранаПолучателя] = 'Литва';
```

В следующем примере создается вычисляемое поле, в котором выводится значение несмещенной дисперсии для значений поля «СтоимостьДоставки» в той же таблице «Заказы». Откройте новую форму и задайте для свойства **Источник записей (RecordSource)** значение «Заказы». Задайте в свойстве поля **Данные (ControlSource)** следующее выражение. Если требуется ограничить отбираемый набор записей, например, провести расчет для заказов, отправленных в Литву, следует задать соответствующее значение свойства формы **Фильтр (Filter)**.

```
=Var([СтоимостьДоставки])
```

Инструкция ALTER TABLE (Microsoft Access), пример

Чтобы выполнить следующие примеры в Microsoft Access, сначала в учебной базе данных «Борей» следует создать новый запрос. Закройте диалоговое окно **Добавление таблицы**, не указывая таблицу или запрос. Переключитесь в режим SQL, вставьте в окно SQL отдельный пример и выполните запрос.

Предупреждение. Эти примеры вносят изменения в учебную базу данных «Борей». Перед началом работы полезно сделать резервную копию этой базы.

В следующем примере в таблицу «Сотрудники» добавляется поле «Оклад», имеющее денежный тип:

```
ALTER TABLE Сотрудники ADD COLUMN Оклад CURRENCY;
```

В следующем примере поле «Оклад» удаляется из таблицы «Сотрудники»:

```
ALTER TABLE Сотрудники DROP COLUMN Оклад;
```

После запуска следующих двух примеров для просмотра измененных отношений выполните команду **Схема данных** из меню **Сервис**.

В следующем примере из таблицы «Заказы» удаляется существующий внешний ключ:

```
ALTER TABLE Заказы DROP CONSTRAINT СотрудникиЗаказы;
```

В следующем примере внешний ключ снова добавляется в таблицу «Заказы»:

```
ALTER TABLE Заказы ADD CONSTRAINT СотрудникиЗаказы FOREIGN KEY
(КодСотрудника) REFERENCES Сотрудники (КодСотрудника);
```

Инструкция CREATE INDEX (Microsoft Access), пример

Чтобы выполнить следующие примеры в Microsoft Access, сначала в учебной базе данных «Борей» следует создать новый запрос. Закройте диалоговое окно **Добавление таблицы**, не указывая таблицу или запрос. Переключитесь в режим SQL, вставьте в окно SQL отдельный пример и выполните запрос.

Предупреждение. Эти примеры вносят изменения в учебную базу данных «Борей». Перед началом работы полезно сделать резервную копию этой базы.

В следующем примере в таблице «Сотрудники» создается индекс, состоящий из двух полей: «ДомашнийТелефон» и «Добавочный»:

```
CREATE INDEX НовыйИндекс ON Сотрудники (ДомашнийТелефон, Добавочный);
```

В следующем примере в таблице «Клиенты» создается индекс по полю «КодКлиента». Никакие две записи не могут содержать в этом поле одинаковое значение, а также значение этого поля не может быть равно **Null**.

```
CREATE UNIQUE INDEX КодКл ON Клиенты (КодКлиента) WITH DISALLOW NULL;
```

Для удаления созданного в предыдущем примере индекса следует воспользоваться инструкцией DROP. В режиме конструктора таблицы невозможно удалить индекс, пока в таблице «Клиенты» не удалено также и отношение. Следующая строка удаляет новый индекс с помощью инструкции DROP:

```
DROP INDEX КодКл ON Клиенты;
```

В следующем примере в связанной таблице ODBC создается индекс. На удаленную базу данных, содержащую таблицу, новый индекс не оказывает никакого воздействия. Следующий пример не будет работать, если его просто вставить в Microsoft Access; сначала необходимо создать связанную таблицу ODBC с именем «ЗаказаноODBC».

```
CREATE UNIQUE INDEX КодЗаказа ON ЗаказаноODBC (КодЗаказа);
```

Инструкция CREATE TABLE, предложение CONSTRAINT (Microsoft Access), пример

Чтобы выполнить следующие примеры в Microsoft Access, сначала в учебной базе данных «Борей» следует создать новый запрос. Закройте диалоговое окно **Добавление таблицы**, не указывая таблицу или запрос. Переключитесь в режим SQL, вставьте в окно SQL отдельный пример и выполните запрос.

Предупреждение. Эти примеры вносят изменения в учебную базу данных «Борей». Перед началом работы полезно сделать резервную копию этой базы.

В следующем примере создается новая таблица с двумя текстовыми полями:

```
CREATE TABLE ПерваяТаблица (Имя TEXT, Фамилия TEXT);
```

В следующем примере создается новая таблица с двумя текстовыми полями и одним полем даты/времени. Из этих трех полей создается уникальный индекс:

```
CREATE TABLE ВтораяТаблица (Имя TEXT,
Фамилия TEXT, ДатаРождения DATETIME,
CONSTRAINT МойИндекс UNIQUE (Имя, Фамилия, ДатаРождения));
```

В следующем примере создается новая таблица с двумя текстовыми полями и числовым полем. Поле SSN делается ключевым полем.

```
CREATE TABLE ТретьяТаблица (Имя TEXT, Фамилия TEXT, SSN INTEGER
CONSTRAINT МойИндекс PRIMARY KEY);
```

Инструкция DROP (Microsoft Access), пример

Чтобы выполнить следующие примеры, следует в таблице «Сотрудники» базы данных «Борей» по любому полю создать индекс с именем «МойИндекс». Затем необходимо создать новую таблицу с именем «Ученики», новый запрос в учебной базе данных «Борей» и закрыть диалоговое окно **Добавление таблицы** , не указывая таблицу или запрос. После этого переключитесь в режим SQL, вставьте в окно SQL отдельный пример и выполните запрос.

Предупреждение. Эти примеры вносят изменения в учебную базу данных «Борей». Перед началом работы полезно сделать резервную копию этой базы.

В следующем примере индекс «МойИндекс» удаляется из таблицы «Сотрудники»:

```
DROP INDEX МойИндекс ON Сотрудники;
```

В следующем примере из базы данных удаляется таблица «Ученики»:

```
DROP TABLE Ученики;
```

Инструкция SELECT, предложение FROM (Microsoft Access), пример

Чтобы выполнить следующие примеры в Microsoft Access, сначала в учебной базе данных «Борей» следует создать новый запрос. Закройте диалоговое окно **Добавление таблицы**, не указывая таблицу или запрос. Переключитесь в режим SQL, вставьте в окно SQL отдельный пример и выполните запрос.

В следующем примере из всех записей таблицы «Сотрудники» выбираются поля «Фамилия» и «Имя»:

```
SELECT Фамилия, Имя FROM Сотрудники;
```

В следующем примере из таблицы «Сотрудники» выбираются все поля:

```
SELECT * FROM Сотрудники;
```

В следующем примере вычисляется число записей с заполненным полем «Индекс» и результирующему полю дается имя «Бирка»:

```
SELECT Count(Индекс) AS Бирка FROM Клиенты;
```

В следующем примере показывается, какова была бы цена товаров, если бы она возросла на 10 процентов. Пример не изменяет существующие в базе данных цены.

```
SELECT Марка, Цена AS Текущий, Цена * 1.1 AS ПредлагаемаяНоваяЦена
FROM Товары;
```

В следующем примере подсчитывается число товаров в базе данных, а также их средняя и максимальная цена:

```
SELECT Count(*) AS [Всего товаров], Avg(Цена) AS [Средняя цена], Max(Цена)
AS [Максимальная цена] FROM Товары;
```

В следующем примере для каждой записи таблицы «Товары» выводятся марка и цена. Строка "по цене" разделяет два поля в результирующем наборе.

```
SELECT Марка, 'по цене', Цена FROM Товары;
```

Предложение IN (Microsoft Access), пример

В следующем примере показано использование предложения IN для получения информации из внешней базы данных. В каждом примере предполагается, что таблица «Клиенты» находится во внешней базе данных.

Примечание. Предложение SQL IN отличается от оператора SQL **In**, используемого для проверки совпадения значения выражения с некоторым значением из заданного списка.

Внешняя база данных	Инструкция SQL

Ядро баз данных Microsoft Jet.
Файл ДанныеJet.mdb является базой данных Jet, содержащей таблицу Клиенты.
	`SELECT КодКлиента`
	`FROM Клиенты`
	`IN 'C:\Мои документы\ДанныеJet.mdb'`
	`WHERE КодКлиента Like 'A*';`

Книга Microsoft Excel (файл .xls).
Файл ДанныеXL.xls является книгой Microsoft Excel, содержащей лист с именем Клиенты. ДиапазонКлиентов – это поименованный диапазон листа. Следует отметить, что для указания в качестве таблицы целого листа, необходимо добавить к нему знак доллара ($) и заключить имя в квадратные скобки.
	`SELECT КодКлиента, Название`
	`FROM [Клиенты$]`
	`IN 'C:\Мои документы\ДанныеXL.xls' 'EXCEL 5.0;'`
	`WHERE КодКлиента Like 'A*';`
	– или –
	`SELECT КодКлиента, Название`
	`FROM ДиапазонКлиентов`
	`IN 'C:\Мои документы\ДанныеXL.xls' 'EXCEL 5.0;'`
	`WHERE КодКлиента Like 'A*';`

dBASE III или IV.
Для извлечения данных из таблицы dBASE III замените dBASE IV на dBASE III.
	`SELECT КодКлиента`
	`FROM Клиенты`
	`IN C:\DBASE\DATA\ПРОДАЖИ dBASE IV;`
	`WHERE КодКлиента Like A*;`

dBASE III или IV (с использованием синтаксиса DATABASE).
Для извлечения данных из таблицы dBASE III замените dBASE IV на dBASE III.
	`SELECT КодКлиента`
	`FROM Клиенты`
	`IN [dBASE IV;DATABASE=C:\DBASE\DATA\ ПРОДАЖИ;]`
	`WHERE КодКлиента Like A*;`

Paradox 3.*x* или 4.*x*.
Для извлечения данных из таблицы Paradox версии 3.*x* замените Paradox 4.*x* на Paradox 3.*x*.
	`SELECT КодКлиента`
	`FROM Клиенты`
	`IN C:\PARADOX\DATA\ПРОДАЖИ Paradox 4.x;`
	`WHERE КодКлиента Like A*;`

Paradox 3.*x* или 4.*x* (с использованием синтаксиса DATABASE).
Для извлечения данных из таблицы Paradox версии 3.*x*, замените Paradox 4.*x* на Paradox 3.*x*.
	`SELECT КодКлиента`
	`FROM Клиенты`
	`IN [Paradox 4.x;DATABASE=C:\PARADOX\DATA\ ПРОДАЖИ;]`
	`WHERE КодКлиента Like A*;`

Оператор In (Microsoft Access), пример

Чтобы выполнить следующий пример в Microsoft Access, сначала в учебной базе данных «Борей» следует создать новый запрос. Закройте диалоговое окно **Добавление таблицы**, не указывая таблицу или запрос. Переключитесь в режим SQL, вставьте в окно SQL отдельный пример и выполните запрос.

В этом примере в таблице «Заказы» создается запрос, возвращающий заказы, отправляемые в Киев, Минск и Тулу:

```
SELECT * FROM Заказы WHERE ГородПолучателя In ('Киев','Минск','Тула');
```

Предложение WHERE (Microsoft Access), пример

Чтобы выполнить следующие примеры в Microsoft Access, сначала в учебной базе данных «Борей» следует создать новый запрос. Закройте диалоговое окно **Добавление таблицы**, не указывая таблицу или запрос. Переключитесь в режим SQL, вставьте в окно SQL отдельный пример и выполните запрос.

В следующем примере выбираются поля «Фамилия» и «Имя» всех записей, значение поля «Фамилия» в которых Иванов:

```
SELECT Фамилия, Имя FROM Сотрудники
WHERE Фамилия = 'Иванов';
```

В следующем примере выбираются поля «Фамилия» и «Имя» для всех сотрудников, чья фамилия начинается на букву С:

```
SELECT Фамилия, Имя FROM Сотрудники
WHERE Фамилия Like 'С*';
```

В следующем примере выбираются товары, чья цена попадает в диапазон от $20 до $50 включительно:

```
SELECT Марка, Цена FROM Товары
WHERE (Цена >=20.00 And Цена <= 50.00);
```

В следующем примере выбираются все продукты, чьи названия попадают в алфавитном порядке в диапазон от «Кол» до «Мол» включительно. Продукт «Молоко» не будет отобран, потому что «Молоко» следует в алфавитном порядке за «Мол» и поэтому не попадает в заданный диапазон.

```
SELECT Марка, Цена FROM Товары
WHERE Марка Between 'Кол' And 'Мол';
```

В следующем примере выбираются заказы, размещенные в первой половине 1995 года:

```
SELECT КодЗаказа, ДатаРазмещения FROM Заказы
WHERE ДатаРазмещения Between #1-1-95# And #6-30-95#;
```

В следующем примере выбираются заказы, предназначенные для штатов Айдахо, Орегон или Вашингтон:

```
SELECT КодЗаказа, ОбластьПолучателя FROM Заказы
WHERE ОбластьПолучателя In ('ID', 'OR', 'WA');
```

Предложение GROUP BY (Microsoft Access), пример

Чтобы выполнить следующие примеры в Microsoft Access, сначала в учебной базе данных «Борей» следует создать новый запрос. Закройте диалоговое окно **Добавление таблицы**, не указывая таблицу или запрос. Переключитесь в режим SQL, вставьте в окно SQL отдельный пример и выполните запрос.

В следующем примере определяется средняя цена всех продуктов каждого поставщика и список цен группируется по поставщикам:

```
SELECT КодПоставщика, Avg(Цена) AS СредняяЦена
FROM Товары GROUP BY КодПоставщика;
```

В следующем примере определяется максимальная цена товара каждого типа:

```
SELECT КодТипа, Max(Цена) AS МаксимальнаяЦена
FROM Товары GROUP BY КодТипа;
```

В следующем примере подсчитывается число заказов, относящихся к каждому сотруднику в базе данных:

```
SELECT КодСотрудника, Count(КодЗаказа) AS КодЧислаЗаказов
FROM Заказы GROUP BY КодСотрудника;
```

Предложение HAVING (Microsoft Access), пример

Чтобы выполнить следующие примеры в Microsoft Access, сначала в учебной базе данных «Борей» следует создать новый запрос. Закройте диалоговое окно **Добавление таблицы**, не указывая таблицу или запрос. Переключитесь в режим SQL, вставьте в окно SQL отдельный пример и выполните запрос.

В следующем примере определяются все поставщики, средняя цена товаров которых превышает $25:

```
SELECT КодПоставщика, Avg(Цена) AS СредняяЦена
FROM Товары GROUP BY КодПоставщика
HAVING (Avg(Цена)>25);
```

В следующем примере отбираются сотрудники, обслужившие более 100 заказов:

```
SELECT КодСотрудника, Count(КодЗаказа) AS КодЧислаЗаказов
FROM Заказы GROUP BY КодСотрудника
HAVING Count(КодЗаказа) > 100;
```

Предложение ORDER BY (Microsoft Access), пример

Чтобы выполнить следующие примеры в Microsoft Access, сначала в учебной базе данных «Борей» следует создать новый запрос. Закройте диалоговое окно **Добавление таблицы**, не указывая таблицу или запрос. Переключитесь в режим SQL, вставьте в окно SQL отдельный пример и выполните запрос.

В следующей инструкции SQL предложение ORDER BY используется для сортировки записей в алфавитном порядке, а затем по типам.

В этом примере записи сортируются по фамилии в убывающем порядке (от Я до А):

```
SELECT Фамилия, Имя FROM Сотрудники ORDER BY Фамилия DESC;
```

В следующем примере выполняется сортировка сначала по коду типа, а затем по марке товара:

```
SELECT КодТипа, Марка, Цена FROM Товары
ORDER BY КодТипа, Марка;
```

Предикаты ALL, DISTINCT, DISTINCTROW и TOP (Microsoft Access), пример

Чтобы выполнить следующие примеры в Microsoft Access, следует создать две новые таблицы, показанные ниже. Затем в учебной базе данных «Борей» следует создать новый запрос и закрыть диалоговое окно **Добавление таблицы**, не указывая таблицу или запрос. После этого переключиться в режим SQL, вставить в окно SQL отдельный пример и выполнить запрос.

Таблица «Клиенты»

Имя	КодКлиента
Александр	1
Алексей	2
Борис	3
Александр	4

Таблица «Счета»

КодКлиента	КодСчета
1	1
1	2
2	3
2	4
2	5
4	6
4	7

Следующий пример возвращает все записи, получающиеся в результате внутреннего объединения двух таблиц. Результирующий набор записей является обновляемым.

```
SELECT ALL Имя FROM Клиенты INNER JOIN Счета
ON Клиенты.КодКлиента = Счета.КодКлиента;
```

Результат	Обновляемый
Александр	Да
Александр	
Алексей	
Алексей	
Алексей	
Александр	
Александр	

В следующем примере отбираются только записи с уникальными значениями, возвращаемыми в результате внутреннего объединения двух таблиц. Результирующий набор записей не является обновляемым.

```
SELECT DISTINCT Имя FROM Клиенты INNER JOIN Счета
ON Клиенты.КодКлиента = Счета.КодКлиента;
```

Результат	Обновляемый
Александр	Нет
Алексей	

В следующем примере отбираются только уникальные записи, возвращаемые в результате внутреннего объединения двух таблиц. Результирующий набор записей является обновляемым

```
SELECT DISTINCTROW Имя FROM Клиенты INNER JOIN Счета
ON Клиенты.КодКлиента = Счета.КодКлиента;
```

Результат	Обновляемый
Александр	Да
Алексей	
Александр	

В следующем примере отбираются первые пять записей, возвращаемые в результате внутреннего объединения двух таблиц. Результирующий набор записей является обновляемым.

```
SELECT TOP 5 Имя FROM Клиенты INNER JOIN Счета
ON Клиенты.КодКлиента = Счета.КодКлиента
ORDER BY Счета.КодСчета;
```

Результат	Обновляемый
Александр	Да
Александр	
Алексей	
Алексей	
Алексей	

Инструкция DELETE (Microsoft Access), пример

Чтобы выполнить следующий пример, в таблицу «Сотрудники» следует добавить несколько записей со значением «Ученик» в поле «Должность».

Этот пример удаляет все записи для работников, чья должность «Ученик». Если предложение FROM включает только одну таблицу, не нужно указывать имя таблицы в инструкции DELETE.

```
DELETE * FROM Сотрудники WHERE Должность = 'Ученик';
```

Оператор INNER JOIN (Microsoft Access), пример

Чтобы выполнить следующий пример в Microsoft Access, сначала в учебной базе данных «Борей» следует создать новый запрос. Закройте диалоговое окно **Добавление таблицы**, не указывая таблицу или запрос. Переключитесь в режим SQL, вставьте в окно SQL отдельный пример и выполните запрос.

В следующем примере создаются два эквисоединения: одно между таблицами «Заказано» и «Заказы», а другое между таблицами «Заказы» и «Сотрудники». Оба соединения необходимы, поскольку таблица «Сотрудники» не содержит сведения о продажах, а таблица «Заказано» – сведения о сотрудниках. В результате выполнения запроса формируется список сотрудников и общего количества их продаж.

```
SELECT DISTINCTROW Sum(Цена * Количество)
AS Продажи, Имя & " " & Фамилия AS ФИО FROM Сотрудники
INNER JOIN(Заказы INNER JOIN [Заказано]
ON Заказы.КодЗаказа = [Заказано].КодЗаказа)
ON Сотрудники.КодСотрудника = Заказы.КодСотрудника
GROUP BY Имя & " " & Фамилия;
```

Инструкция INSERT INTO (Microsoft Access), пример

Предупреждение. Эти примеры вносят изменения в учебную базу данных «Борей». Перед началом работы полезно сделать резервную копию этой базы.

Чтобы выполнить следующий пример, следует создать копию таблицы «Клиенты» и назвать ее «НовыеКлиенты». В этом примере из таблицы «НовыеКлиенты» выбираются все записи и затем добавляются в таблицу «Клиенты». Когда отдельные столбцы не указаны, имена столбцов таблицы в инструкции SELECT должны в точности совпадать с именами столбцов таблицы в предложении INSERT INTO.

```
INSERT INTO Клиенты SELECT * FROM НовыеКлиенты;
```

В следующем примере в таблице «Сотрудники» создается новая запись:

```
INSERT INTO Сотрудники (Имя, Фамилия, Должность) VALUES ('Борис',
'Колесников', 'Ученик');
```

В следующем примере из предполагаемой таблицы «Ученики» отбираются все ученики, нанятые более 30 дней назад, и их записи добавляются в таблицу «Сотрудники».

```
INSERT INTO Сотрудники SELECT Ученики.* FROM Ученики WHERE ДатаНайма <
Now() - 30;
```

Предложения INSERT INTO может использоваться для сохранения копии сведений базы данных перед их изменением. Например, можно сохранить сведения из таблицы «Сотрудники» перед изменением пользователем данных в ней.

Чтобы выполнить следующий пример, следует в окне базы данных скопировать таблицу «Сотрудники» в новую таблицу «ИсторияСотрудников» путем выделения таблицы, нажатия кнопки панели инструментов **Копировать**  , а затем кнопки **Вставить**  . В группе **Параметры вставки** диалогового окна **Вставка таблицы** следует выбрать флажок **Только структура**. В результате этого будет скопирована только структура, не включая данные. Затем необходимо вставить следующую инструкцию SQL в новый запрос и сохранить его под именем «ЗапросРезервирования»:

```
INSERT INTO ИсторияСотрудников (Имя, Фамилия, Должность)
VALUES (Forms!Сотрудники!Имя, Forms!Сотрудники!Фамилия,Forms!Сотрудники!
Должность);
```

Откройте форму Сотрудники в режиме конструктора формы и установите значение свойства **До обновления (BeforeUpdate)** равным [Процедура обработки события]. В процедуру обработки события **До обновления (BeforeUpdate)** добавьте следующие строки:

```
DoCmd.OpenQuery "ЗапросРезервирования"
```

Теперь каждый раз при внесении изменений существующие в таблице «Сотрудники» данные будут копироваться в запасную таблицу.

Операции LEFT JOIN, RIGHT JOIN (Microsoft Access), пример

Чтобы выполнить следующие примеры в Microsoft Access, сначала в учебной базе данных «Борей» следует создать новый запрос. Закройте диалоговое окно **Добавление таблицы**, не указывая таблицу или запрос. Переключитесь в режим SQL, вставьте в окно SQL отдельный пример и выполните запрос.

В следующем примере отбираются все клиенты, включая тех, кто не имеет ни одного заказа:

```
SELECT Клиенты.КодКлиента, Название, КодЗаказа
FROM Клиенты
LEFT JOIN Заказы
ON Клиенты.КодКлиента = Заказы.КодКлиента
ORDER BY КодЗаказа;
```

В следующем примере отбираются все поставщики, включая тех, кто не поставил ни одного товара. Перед выполнением этого пример следует запустить следующий запрос, который добавит запись в таблицу «Поставщики»:

```
INSERT INTO Поставщики (Название, ОбращатьсяК, Должность)
VALUES ('ООО Экзотика', 'Вероника Кудрявцева', 'Менеджер по закупкам');
```

Затем необходимо создать новый запрос, в окне SQL вставить в него следующую инструкцию SQL и выполнить.

```
SELECT Поставщики.КодПоставщика, Поставщики.Название, Товары.КодТовара,
Товары.Марка
FROM Товары RIGHT JOIN Поставщики ON Товары.КодПоставщика =
Поставщики.КодПоставщика
ORDER BY Товары.КодПоставщика;
```

Чтобы удалить созданные лишние записи из таблицы «Поставщики», следует выполнить следующий запрос:

```
DELETE * FROM Поставщики
WHERE Название = 'ООО Экзотика';
```

Описание PARAMETERS (Microsoft Access), пример

Чтобы выполнить следующие примеры в Microsoft Access, сначала в учебной базе данных «Борей» следует создать новый запрос. Закройте диалоговое окно **Добавление таблицы**, не указывая таблицу или запрос. Переключитесь в режим SQL, вставьте в окно SQL отдельный пример и выполните запрос.

В следующем примере пользователю предлагается указать фамилию сотрудника, и затем эта фамилия используется в качестве условия для запроса:

```
PARAMETERS [Введите фамилию:] Text;
SELECT *
FROM Сотрудники
WHERE Фамилия = [Введите фамилию:];
```

В следующем примере пользователю предлагается указать код типа, и затем это значение используется в качестве условия для запроса:

```
PARAMETERS [Введите код типа:] Value;
SELECT КодТипа, Название, Count([Заказано].КодЗаказа) AS Бирка
FROM Товары
INNER JOIN [Заказано] ON Товары.КодТовара = [Заказано].КодТовара
GROUP BY КодТипа, Марка
HAVING КодТипа = [Введите код типа:];
```

Инструкция SELECT...INTO (Microsoft Access), пример

Чтобы выполнить следующие примеры в Microsoft Access, сначала в учебной базе данных «Борей» следует создать новый запрос. Закройте диалоговое окно **Добавление таблицы**, не указывая таблицу или запрос. Переключитесь в режим SQL, вставьте в окно SQL отдельный пример и выполните запрос.

В следующем примере из таблицы «Сотрудники» отбираются все записи, а затем копируются в новую таблицу с именем «КопияСотрудников»:

```
SELECT * INTO [КопияСотрудников] FROM Сотрудники;
```

В следующем примере создается новая таблица «Представители», в которую помещаются только записи о представителях по продажам:

```
SELECT Сотрудники.Имя, Фамилия INTO [Представители]
FROM Сотрудники
WHERE Должность = 'Представитель по продажам';
```

В следующем примере создается копия таблицы «Сотрудники» и затем помещается предполагаемую базу данных Копия.mdb:

```
SELECT Сотрудники.* INTO Сотрудники IN Копия.mdb FROM Сотрудники;
```

В следующем примере предполагается, что существует таблица «ПлатежнаяВедомость» с двумя полями: «КодСотрудника» и «Оклад». Далее создается новая таблица, содержащая коды и оклады всех учеников. Между таблицами «Сотрудники» и «ПлатежнаяВедомость» устанавливается отношение один к одному. Новая таблица содержит все данные из таблицы «Сотрудники», а также поле «Оклад» из таблицы «ПлатежнаяВедомость».

```
SELECT Сотрудники.*, Оклад INTO Ученики
FROM Сотрудники
INNER JOIN ПлатежнаяВедомость ON Сотрудники.КодСотрудника =
ПлатежнаяВедомость.КодСотрудника
WHERE Должность = 'Ученик';
```

Инструкция TRANSFORM (Microsoft Access), пример

Чтобы выполнить следующие примеры в Microsoft Access, сначала в учебной базе данных «Борей» следует создать новый запрос. Закройте диалоговое окно **Добавление таблицы**, не указывая таблицу или запрос. Переключитесь в режим SQL, вставьте в окно SQL отдельный пример и выполните запрос.

В следующем примере создается перекрестный запрос, отражающий продажу товаров по месяцам за указанный пользователем год. Месяцы представляются в виде столбцов, а марки товаров в виде строк.

```
PARAMETERS [Продажи за какой год?] LONG;
TRANSFORM Sum([Заказано].Количество * ([Заказано].Цена - ([Заказано].Скидка
/ 100) * [Заказано].Цена)) AS Продажи
SELECT Марка FROM Заказы
INNER JOIN (Товары INNER JOIN [Заказано] ON Товары.КодТовара =
[Заказано].КодТовара)
ON Заказы.КодЗаказа = [Заказано].КодЗаказа
WHERE DatePart("yyyy", ДатаРазмещения) = [Продажи за какой год?]
GROUP BY Марка
ORDER BY Марка
PIVOT DatePart("m", ДатаРазмещения);
```

В следующем примере создается перекрестный запрос, отражающий продажу товаров каждым поставщиком по кварталам указанный пользователем год. Кварталы представляются в виде столбцов, а имена поставщиков в виде строк.

```
PARAMETERS [Продажи за какой год?] LONG;
TRANSFORM Sum([Заказано].Количество * ([Заказано].Цена - ([Заказано].Скидка
/ 100) * [Заказано].Цена)) AS Продажи
SELECT НазваниеПолучателя FROM Заказы
INNER JOIN ((Поставщики INNER JOIN Товары ON Поставщики.КодПоставщика =
Товары.КодПоставщика)
INNER JOIN [Заказано] ON Товары.КодТовара = [Заказано].КодТовара)
ON Заказы.КодЗаказа = [Заказано].КодЗаказа
WHERE DatePart("yyyy", ДатаРазмещения) = [Продажи за какой год?]
GROUP BY Название
ORDER BY Название
PIVOT "Qtr " & DatePart("q", ДатаРазмещения) In ('Квартал 1', 'Квартал 2',
'Квартал 3', 'Квартал 4');
```

Инструкция UPDATE (Microsoft Access), пример

Чтобы выполнить следующие примеры в Microsoft Access, сначала в учебной базе данных «Борей» следует создать новый запрос. Закройте диалоговое окно **Добавление таблицы**, не указывая таблицу или запрос. Переключитесь в режим SQL, вставьте в окно SQL отдельный пример и выполните запрос.

Предупреждение. Эти примеры вносят изменения в учебную базу данных «Борей». Перед началом работы полезно сделать резервную копию этой базы.

В следующем примере значение поля «Подчиняется» изменяется на 5 для всех работников, в записях которых это поле было равно 2:

```
UPDATE Сотрудники SET Подчиняется = 5 WHERE Подчиняется = 2;
```

В следующем примере для поставщика номер 8 цена товаров, поставки которых не прекращены, увеличивается на 10 процентов:

```
UPDATE Товары SET Цена = Цена * 1.1
WHERE КодПоставщика = 8 AND ПоставкиПрекращены = No;
```

В следующем примере на 5 процентов уменьшается цена товаров, поставки которых не прекращены, для поставщика «Tokyo Traders». Между таблицами «Товары» и «Поставщики» устанавливается отношение многие к одному.

```
UPDATE Поставщики INNER JOIN Товары
ON Поставщики.КодПоставщика = Товары.КодПоставщика SET Цена = Цена * .95
WHERE Название = 'Tokyo Traders' AND ПоставкиПрекращены = No;
```

Операция UNION (Microsoft Access), пример

Чтобы выполнить следующие примеры в Microsoft Access, сначала в учебной базе данных «Борей» следует создать новый запрос. Закройте диалоговое окно **Добавление таблицы**, не указывая таблицу или запрос. Переключитесь в режим SQL, вставьте в окно SQL отдельный пример и выполните запрос.

В следующем примере выводятся названия и города всех поставщиков и клиентов из Бразилии:

```
SELECT Название, Город FROM Поставщики
WHERE Страна = 'Бразилия'
UNION SELECT Название, Город FROM Клиенты
WHERE Страна = 'Бразилия';
```

В следующем примере выводятся названия и города всех поставщиков и клиентов, находящихся в Бразилии:

```
SELECT Название, Город, 'Поставщик' AS Источник
FROM Поставщики
WHERE Страна = 'Бразилия'
UNION SELECT Название, Город, 'Клиент'
FROM Клиенты
WHERE Страна = 'Бразилия'
ORDER BY Город, Страна;
```

В следующем примере выводятся названия и коды всех поставщиков и клиентов. Такое объединение предполагает, что таблицы имеют одинаковое число столбцов.

```
TABLE Клиенты UNION TABLE Поставщики;
```

Подчиненные запросы SQL (Microsoft Access), пример

Чтобы выполнить следующие примеры в Microsoft Access, сначала в учебной базе данных «Борей» следует создать новый запрос. Закройте диалоговое окно **Добавление таблицы**, не указывая таблицу или запрос. Переключитесь в режим SQL, вставьте в окно SQL отдельный пример и выполните запрос.

В следующем примере выводятся все заказы без скидок, сумма которых превышает среднюю сумму заказов:

```
SELECT КодЗаказа, (Цена * Количество) As СуммаЗаказа FROM [Заказано]
WHERE Скидка = 0 AND (Цена * Количество) > ALL(SELECT Avg(Цена *
Количество)
FROM [Заказано]);
```

В следующем примере выводятся марки и цены всех товаров, цена которых совпадает с ценой анисового сиропа:

```
SELECT Марка, Цена FROM Товары
WHERE Цена = (SELECT Цена FROM [Товары]
WHERE Марка = 'Анисовый сироп');
```

В следующем примере выводятся названия и представители всех клиентов, размещавших заказы во втором квартале 1995 года:

```
SELECT ОбращатьсяК, Название, Должность, Телефон FROM Клиенты
WHERE КодКлиента IN (SELECT КодКлиента FROM Заказы WHERE ДатаРазмещения
BETWEEN #04/1/95# AND #06/30/95#);
```

В следующем примере выводятся имена всех сотрудников, зарегистрировавших хотя бы один заказ. То же самое можно сделать используя предложение INNER JOIN.

```
SELECT Имя, Фамилия FROM Сотрудники
WHERE EXISTS (SELECT КодЗаказа FROM Заказы
WHERE Заказы.КодСотрудника = Сотрудники.КодСотрудника);
```

Оператор Like (Microsoft Access), пример

В следующем примере выводятся фамилии сотрудников, начинающиеся с букв от «А» до «Г». Чтобы выполнить следующий пример в Microsoft Access, сначала в учебной базе данных «Борей» следует создать новый запрос. Закройте диалоговое окно **Добавление таблицы**, не указывая таблицу или запрос. Переключитесь в режим SQL, вставьте в окно SQL отдельный пример и выполните запрос.

```
SELECT * FROM Сотрудники WHERE Фамилия Like '[А-Г]*';
```

Модуль, содержащий инструкцию Option Private, не может быть модулем класса (Microsoft Access)

В Microsoft Access модули форм и модули отчетов являются модулями класса. Включение инструкции **Option Private Module** в модуль формы или отчета не допускается.

Директива #Const (условные константы компилятора) (Microsoft Access)

В Microsoft Access пользователь имеет возможность определять общие условные константы компилятора в разделе описаний модуля. Общие константы компилятора могут быть использованы во всех модулях в текущей базе данных. В других базах данных эти константы использовать нельзя.

Допускается также описание общей константы компилятора в окне диалога **Параметры**. Для этого следует выбрать команду **Параметры** в меню **Сервис,** выбрать вкладку **Другие** и ввести выражение, определяющее константу, в поле **Аргументы условной компиляции**.

Например, можно ввести в поле **Аргументы условной компиляции** следующее выражение:

```
conDebug = 1
```

После этого становится возможным использование общей константы компилятора в программе в лбом модуле. Например, включение этой константы в конструкцию **#If...Then...#Else** делает возможным условный запуск программы в режиме отладки.

```
#If conDebug = 1 Then
    .      ' Выполнение программы в режиме отладки.
    .
    .
#Else
    .      ' Выполнение программы в стандартном режиме.
    .
    .
#End If
```

При одновременном создании нескольких общих условных констант компилятора необходимо описывать их на отдельных сроках в разделе описаний. В поле **Аргументы условной компиляции** следует разделять их описания двоеточием. Например, правильным является описание в поле **Аргументы условной компиляции** следующих констант:

```
conActiveLanguage = 1 : conDebug = 1
```

Для включения в конструкцию **#If...Then...#Else** обеих констант может использоваться логический оператор. В приведенном ниже примере первый блок программы выполняется, только если обе константы равны 1 и, таким образом, логическое выражение является истинным.

```
#If (conActiveLanguage = 1 And conDebug = 1) Then
    .      ' Выполнение блока отладки для версии активного языка.
    .
    .
#Else
    .      ' Выполнение другого блока программы.
    .
    .
#End If
```

Если одна или обе константы равны 0, выполняется блок программы в **#Else** части конструкции. Для изменения выполняемого блока достаточно изменить значение констант.

Примечания

- Константа условной компиляции всегда проверяется с помощью метода сравнения строк. Этот способ сравнения эквивалентен включению в модуль инструкции **Option Compare Text**. Такой способ сравнения применяется даже в том случае, когда модуль содержит инструкцию

Option Compare Database.

- При разработке программы, предназначенной для условной компиляции, иногда удобнее просматривать процедуры по одной, а не все сразу. Для изменения способа просмотра программ выберите в меню **Сервис** команду **Параметры** и выберите вкладку **Модуль**. В группе **Просмотр программ** снимите флажок **Полный модуль**.

Функция Choose (Microsoft Access)

Функция **Choose** также используется в вычисляемом элементе управления в форме или отчете Microsoft Access. Например**,** функция **Choose** позволяет задать значение элемента управления в зависимости от значения другого поля. Для этого в ячейку свойства **Данные (ControlSource)** элемента управления следует ввести выражение, содержащее функцию **Choose**. Выражение в следующем примере может использоваться для задания свойства **ControlSource** элемента управления на основе значения поля «Доставка» в таблице «Заказы».

```
=Choose([Доставка], "Иное", "Ространс", "Почта")
```

В приведенном примере, если поле «Доставка» имеет значение 1, функция **Choose** возвращает первую строку, «Иное». Если поле «СтоимостьДоставки» имеет значение 2, возвращается вторая строка и т.д. Если поле "Доставка" имеет значение 0, то функция **Choose** возвращает пустое значение (**Null**).

В следующем примере демонстрируется, как обеспечить возвращение конкретного значения, даже если поле содержит 0.

```
= Choose(СтоимостьДоставки + 1, "Нет", "Иное", "Ространс", "Почта")
```

Примечание. В модулях Visual Basic для возвращения значения, выбираемого из набора значений, обычно используют более гибкую конструкцию **Select Case**.

Функция IIf (Microsoft Access)

Функцию **IIf** (немедленное ЕСЛИ) также используют в вычисляемом элементе управления в форме или отчете Microsoft Access. Функция **IIf** находит значение выражения и возвращает одно из двух значений в зависимости от того, имеет ли это выражение значение **True** (-1) или **False** (0). Например, функция **IIf** позволяет проверить поле в форме и определить, имеет ли данное поле пустое (**Null**) значение. Если поле пустое, то с помощью функции **IIf** можно возвратить пустую строку, а для непустых значений возвратить значение поля.

В следующем примере проверяется поле «СтранаПолучателя» в таблице «Заказы», и если оно имеет значение **Null**, возвращается пустая строка.

```
= IIf(IsNull(Forms!Заказы![СтранаПолучателя]), "", Forms!Заказы!
[СтранаПолучателя])
```

В конструкциях Visual Basic при вызове функции **IIf** определяются значения обоих аргументов, *значениеИстина* и *значениеЛожь*, хотя возвращается только одно из них. В формах и отчетах Microsoft Access, однако, при вызове функции **IIf** рассчитывается только одно из значений, *значениеИстина* или *значениеЛожь*, именно то, которое возвращается. Таким образом, при использовании функции **IIf** в вычисляемом элементе управления, в макросе или в условном выражении в запросе можно не беспокоиться о возможных побочных эффектах, связанных с вычислением ненужного значения.

Примечание. Функция Microsoft Access **Nz** преобразует пустые значения в нули, в пустые строки или в другие значения, указанные пользователем. В выражениях, предназначенных для обработки значений **Null**, иногда удобнее использовать вместо функции **IIf** функцию **Nz**.

Функцию **IIf** полезно использовать в выражениях в формах и отчетах, а не в программах Visual Basic. В программах Visual Basic более гибкой является полномасштабная конструкция **If...Then...Else**.

Функция Switch (Microsoft Access)

Функция **Switch** также используется в вычисляемом элементе управления в форме или отчете Microsoft Access. Например, функция **Switch** позволяет задать значение элемента управления в зависимости от значения другого поля. Для этого в ячейку свойства **Данные (ControlSource)** элемента управления следует ввести выражение, содержащее функцию **Switch**.

В следующем примере функция **Switch** возвращает строку, определяющую способ доставки товара в зависимости от значения поля «СтоимостьДоставки». Если значение поля «СтоимостьДоставки» меньше 25000р., функция **Switch** возвращает строку «Почта»; если стоимость лежит в интервале от 25000р. до 50000р., возвращается «Ространс»; а для значений, превышающих 50000р., **Switch** возвращает «Иное».

```
= Switch([СтоимостьДоставки] < 25000, "Почта",
  ([СтоимостьДоставки] >= 25000 and [СтоимостьДоставки] <= 50000),
  "Ространс", [СтоимостьДоставки] > 50000, "Иное")
```

Примечание. В программах Visual Basic для возвращения значения, выбираемого из набора значений, обычно используют более гибкую конструкцию **Select Case**.

Функция Fix (Microsoft Access)

При использовании функции **Fix** для добавления нового поля в запрос на создание таблицы, ему будет присвоен тип данных **dbDouble** (с плавающей точкой, 8 байт). Тип данных для поля задается с помощью свойства объектов доступа к данным (DAO) **Type**.

Функция Choose (Microsoft Access), пример

Функция **Choose** позволяет создать вычисляемый элемент управления, значение которого определяется значением поля таблицы базы данных. Предположим, например, что имеется таблица «Доставка» , содержащая поле «КодДоставки». В этом случае можно создать в форме вычисляемое поле и вывести в нем название вида доставки на основании значения поля «КодДоставки».

```
=Choose([Доставка], "Самовывоз", "Совтрансмото", "Почта России")
```

Функция DateAdd (Microsoft Access), пример

В следующем примере демонстрируется использование функции **DateAdd** для создания вычисляемого поля, в котором будет выводиться дата отправки заказа – в данном случае, дата, отстоящая на 30 дней от значения поля «ДатаРазмещения». Подразумевается, что базовой таблицей формы является таблица «Заказы», содержащая поле «ДатаРазмещения». Вычисляемое поле, в котором будет выводиться дата отгрузки, определяется с помощью следующего выражения, вводящегося в ячейку свойства **Данные (ControlSource)** этого поля:

```
= DateAdd("d", 30, [ДатаРазмещения])
```

Функция DateDiff (Microsoft Access), пример

В данном примере функция **DateDiff** используется для расчета числа календарных недель в интервале от первого дня года до текущей даты, а также для расчета числа дней в интервале от 1 февраля 1995 г. до текущей даты.

```
Debug.Print DateDiff("ww", "1-1", Now())
Debug.Print DateDiff("y", #1-фев-1995#, Now())
```

Следующий пример демонстрирует использование функции **DateDiff** в выражении в запросе. Предположим, например, что имеется таблица «Заказы», содержащая поля «ДатаРазмещения» и «ДатаИсполнения». В этом случае можно создать в запросе вычисляемое поле, в котором будет выводиться число дней, прошедших между размещением и выполнением каждого заказа. Для этого следует открыть новый запрос в окне запроса, включить в него таблицу «Заказы» и переместить с помощью мыши поле «Заказ» в бланк запроса по образцу. Для создания вычисляемого поля введите в пустую ячейку в строке «Поле» следующее выражение:

```
СрокИсполнения: DateDiff("y", [ДатаРазмещения], [ДатаИсполнения])
```

Функция DatePart (Microsoft Access), пример

В следующем примере с помощью функции **DatePart** задаются условия отбора в запросе на выборку. Предположим, например, что требуется создать запрос по таблице «Заказы», в котором будет выводиться список всех заказов, размещенных в первом квартале 1996 г. Подразумевается, что в таблице «Заказы» имеется поле «Заказ» и поле «ДатаРазмещения». Поле «Заказ» следует поместить в первую ячейку бланка запроса, а в ячейку строки «Условие» под эти полем ввести следующее выражение:

```
(DatePart("q", [ДатаРазмещения]) = 1) and (DatePart("yyyy",
[ДатаРазмещения]) = 1996)
```

Функция Iif (Microsoft Access), пример

В данном примере функция **Iif** проверяет значение поля «СуммаЗаказа» возвращает строку «Крупный», если сумма превышает 1000; в противном случае возвращается строка «Мелкий». В ячейку свойства **Данные (ControlSource)** вычисляемого поля вводится следующее выражение:

```
= IIf([СуммаЗаказа] > 1000, "Крупный", "Мелкий")
```

Функция Partition (Microsoft Access), пример

Функцию **Partition** используют как в бланке запроса по образцу, так и в окне запроса в режиме SQL. В бланке запроса по образцу допускается использование функции **Partition** как для определения вычисляемого поля, так и для указания условий отбора в запросе на выборку.

В данном примере демонстрируется использование функции **Partition** для создания вычисляемого поля, в котором будет выводиться количество записей, попадающих в каждый из указанных диапазонов. Предположим, например, что имеется таблица «Заказы», содержащая поле «СтоимостьДоставки». В окне запроса создайте новый итоговый запрос. Для этого включите в запрос таблицу «Заказы» и нажмите кнопку **Групповые операции** на панели команд окна запроса. Переместите с помощью мыши поле «СтоимостьДоставки» в первую ячейку строки **Поле** бланка запроса и выберите **Count** в ячейке строки **Групповая операция**. В другую ячейку строки **Поле** введите следующее выражение:

```
Диапазон: Partition([СтоимостьДоставки], 0, 1000, 50)
```

Выберите **Группировка** в ячейке строки **Групповая операция** под этим полем и выполните запрос. Функция **Partition** возвращает одиннадцать диапазонов (0:99, 100:199, 200:299 и т.д.). В запросе будет выведено количество заказов, стоимость доставки которых попадает в каждый из указанных диапазонов.

В следующем примере демонстрируется использование функции **Partition** в инструкциях SQL. Создается перекрестный запрос, в котором проверяется значение поля «СуммаЗаказа» в таблице «Заказы». В запросе подсчитывается число заказов каждого клиента, попадающих в каждый из указанных диапазонов. Диапазоны определятся аргументами функции **Partition**: *начало* = 0, *конец* = 1000, *интервал* = 100.

В режиме SQL введите следующие инструкции. При выполнении запроса в заголовках столбцов появятся все диапазоны.

```
TRANSFORM Count(Заказы.КодЗаказа) AS [ПодсчетЗаказов]
SELECT Заказы.КодКлиента
FROM Заказы
GROUP BY Заказы.КодКлиента
PIVOT Partition(Int([СуммаЗаказа]), 0, 1000, 50);
```

Функция Switch (Microsoft Access), пример

В следующем примере с помощью функции **Switch** выбирается язык для оформления заказов по значениям в полях «СтранаПолучателя» или «ГородПолучателя» в таблице «Заказы». Для этого вычисляемое поле определяется с помощью следующего выражения. Это выражение для удобства чтения записано в несколько строк. Допустимым был бы также ввод этого выражения в виде одной строки.

```
= Switch([ГородПолучателя] = "Минск", "Белорусский ", _
   [ГородПолучателя] = "Киев", "Украинский", _
   [ГородПолучателя] = "Рига", "Латвийский", _
   [СтранаПолучателя] = "Литва", "Литовский", _
   True, "Русский")
```

Если город получателя - Минск, функция **Switch** возвращает «Белорусский», если Киев - «Украинский» и т. д. Если город получателя не перечислен в списке, но страна получателя - Литва, будет возвращено значение «Литовский». Если рассматриваемого города нет в списке, функция **Switch** вернет «Русский».

Метод Add (Microsoft Access)

Элементы**,** добавляемые в определяемое пользователем семейство, автоматически индексируются. Данный индекс может быть использован для ссылки на конкретный элемент семейства. Предположим, например, что имеется семейство `colThings`, содержащее четыре объекта. Для ссылки на третий элемент семейства следует использовать выражение `colThings(3)`.

Примечание. При добавлении элементов в объект **Collection** (Семейство) осуществляется их автоматическая индексация, начинающаяся с номера 1. При перечислении элементов объекта **Collection** необходимо помнить, что отсчет индекса начинается с 1. Этим определяемые пользователем семейства отличатся от встроенных семейств, в которых нумерация обычно начинается с 0.

При добавлении элемента в определяемое пользователем семейство существует также возможность в дополнение к автоматическому индексу указать специальный ключ. После этого становятся возможными ссылки на этот элемент с помощью специального ключа. Например, можно добавить в семейство объект `objMine` со следующим ключом

```
colThings.Add Item := objMine, key := ("A")
```

В результате, становятся возможными ссылки на этот объект с помощью выражения `colThings(A)` или с помощью его индекса.

Объект Collection (Microsoft Access)

В Microsoft Access объект **Collection** (семейство) может содержать набор объектов любого типа. Допускается создание объектов **Collection,** содержащих несколько объектов одного или разных типов.

В семействе, определяемом объектом **Collection**, может содержаться любая комбинация объектов Microsoft Access, объектов доступа к данным и объектов других приложений, выводящихся в Microsoft Access. Кроме того, можно добавлять объект **Collection** в семейство, определяемое другим объектом **Collection**.

Допускается также включение в семейство объектов, определяемых пользователем, создаваемых в Microsoft Access. Например, пользователь имеет возможность создавать собственные объекты с помощью специальных процедур и свойств в модуле формы или модуле отчета. После этого становится возможным добавление созданных объектов в семейство (объект **Collection**) и работа с ними как с единым набором.

Создание объектных переменных (Microsoft Access)

В Microsoft Access существует возможность описания объектной переменной особого типа, типа **Object** или **Variant**. Особые объектные типы включают объекты Microsoft Access, объекты доступа к данным (DAO), объекты **Collection** Visual Basic, экземпляры модулей классов и объекты программирования. В следующем примере приведены различные допустимые особые объектные типы.

```
Dim frmOrderForm As Form     ' Объект типа Form.
Dim tdfOrders As TableDef          ' Объект типа TableDef.
Dim colThings As New Collection    ' Новый объект типа Collection.
Dim obj As New Class1        ' Новый экземпляр модуля класса.
Dim appXL As Excel.Application     ' Объект программирования.
```

Примечание. При использовании в описании объектной переменной ключевого слова **New** создается новый объект, а переменной автоматически присваивается указатель на него. В этом случае не требуется использовать инструкцию **Set**.

В модуле формы или модуле отчета для ссылки на форму или отчет, связанный с данным модулем, может использоваться ключевое слово **Me**. Оно позволяет сослаться на текущую форму без указания полной ссылки.

Например для изменения цвета области данных формы в стандартном модуле можно создать следующую процедуру:

```
Sub ChangeFormColor(frmCurrent As Form)
   frmCurrent.Section(acDetail).BackColor = RGB(Rnd * 256, _
      Rnd * 256, Rnd * 256)
End Sub
```

Эту процедуру можно вызывать из модуля формы, задавая в качестве аргумента ключевое слово **Me**. Это слово представляет собой форму, из которой была вызвана процедура, и над которой выполняется операция.

Например, эта процедура может быть вызвана из обработчика события **Нажатие кнопки (Click)** в области формы. Для этого свойству области данных **Нажатие кнопки (OnClick)** следует присвоить значение [Процедура обработки события] и в модуле формы создать следующую процедуру **Sub**:

```
Private Sub Сведения_Click()
   ChangeFormColor Me
End Sub
```

Если форма носит имя «Заказы», эта процедура может быть вызвана и так:

```
ChangeFormColor Forms!Заказы
```

Преимущество использования ключевого слова **Me** состоит в том, что не требуется знать имя формы, чтобы сослаться на нее.

Программирование объектов (Microsoft Access)

В Microsoft Access объект программирования может быть создан либо с помощью ключевого слова **New,** использованного при описании объектной переменной, либо с помощью функций **CreateObject** или **GetObject**.

Microsoft Access поддерживает два различных способа программирования объектов. Microsoft Access является компонентом ActiveX, т. е. существует возможность доступа к объектам, предоставленным для использования другими приложениями, и работа с ними из Microsoft Access, а объекты Microsoft Access могут управляться другими приложениями, поддерживающими программирование объектов.

Примечание. Перед использованием Microsoft Access или другого приложения в качестве компонента необходимо определить ссылку на объектную библиотеку этого приложения. Более подробные сведения об определении ссылок можно получить, выполнив в предметном указателе справочной системы поиск строки «определение ссылок».

Например, при использовании Microsoft Access в качестве компонента возможна работа с объектами Microsoft Excel. Для этого в Microsoft Access следует создать новый объект Microsoft Excel и присвоить объектной переменной ссылку на него. После этого, как и для любого другого объекта, становится доступным изменение его свойств и вызов его процедур. Новый экземпляр класса Microsoft Excel **Application** может быть создан двумя способами.

```
Dim appXL As New Excel.Application

' Или

Dim appXL As Excel.Application
Set appXL = CreateObject("Excel.Application")
```

Если Microsoft Excel уже запущен, для получения ссылки на текущий экземпляр класса **Application** Microsoft Excel можно воспользоваться функцией **GetObject**.

```
Dim appXL As Excel.Application
Set appXL = GetObject( , "Excel.Application")
```

Более подробные сведения об объекте **Application** можно получить, выполнив в предметном указателе справочной системы поиск строки «**Application** - объект».

При использовании Microsoft Access в качестве компонента, возможна работа с его объектами из других приложений. Например, в Microsoft Excel можно открыть форму Microsoft Access и использовать ее для просмотра книги Microsoft Excel. В следующем примере из Microsoft Excel создается новый объект Microsoft Access.

```
Dim appAccess As New Access.Application
```

Если Microsoft Access уже запущен, для получения ссылки на текущий экземпляр класса **Application** Microsoft Access можно воспользоваться функцией **GetObject**.

Избежание конфликтов имен (Microsoft Access)

Ниже перечислены некоторые дополнительные ошибки, связанные с конфликтом имен, встречающиеся в Microsoft Access.

Процедуры и переменные в стандартных модулях и модулях классов

Если в Microsoft Access и стандартный модуль и модуль класса содержат общую процедуру с одинаковым именем, при вызове этой процедуры без указания ее местонахождения будет всегда выполняться процедура из стандартного модуля. Для вызова процедуры из модуля класса необходимо явно указать это имя класса. То же самое справедливо и для переменных с совпадающими именами, описанных в стандартном модуле и модуле класса.

Например, если в модуле формы «Заказы» существует функция с именем CountOrders, эта функция может быть вызвана следующим образом:

```
Form_Заказы.CountOrders
```

Переменные с совпадающими именами, но различной областью определения

Если в процедуре или модуле описаны переменная уровня модуля и переменная уровня процедуры с совпадающими именами, перед именем переменной уровня модуля необходимо указывать имя этого модуля. Это относится к стандартным модулям и модулям классов.

Например, если переменная с именем `varScope` описана на уровне модуля и на уровне процедуры в стандартном модуле «Важные функции», обратиться к переменной уровня модуля можно, использовав синтаксис `[Важные функции].varScope`.

Следует избегать использования переменных с одинаковыми именами в одном модуле.

Имена констант, переменных и процедур в модулях классов

Константе, переменной или процедуре в форме или отчете невозможно присвоить имя, совпадающее с именем метода или свойства этой формы или отчета. Например, при создании процедуры **Sub** с именем «Name» Microsoft Access выдаст ошибку компиляции, поскольку в форме уже существует свойство **Имя (Name)**.

Область определения и область видимости (Microsoft Access)

Переменные и константы, описанные внутри процедуры, имеют область определения на уровне процедуры. Область определения на уровне общего или личного модуля имеют переменные, константы, определяемые пользователем типы и инструкции **Declare,** помещенные в начале модуля. То же относится к процедурам **Sub, Function, Property Get**, **Property Let** и **Property Set**, описанным внутри этого модуля. Более подробные сведения можно получить, выполнив в предметном указателе справочной системы поиск строки «Declare - инструкция» или «Процедуры обработки свойств».

Следующая таблица показывает области определения различных элементов на двух уровнях определения.

Уровень определения	Тип элементов	Область определения	Примечания
На уровне процедуры	Переменные	Личная	
	Константы	Личная	
На уровне модуля	Переменные	Личная, если при описании не использовано ключевое слово **Public**	
	Константы	Личная, если перед именем не использовано ключевое слово **Public**	Нельзя описать общую константу внутри модуля класса.
	Инструкции **Declare**	Общая, если перед именем не использовано ключевое слово **Private**	Перед инструкцией **Declare** в модуле класса должно использоваться ключевое слово **Private**.
	Определяемые пользователем типы	Общая, если перед именем не использовано ключевое слово **Private**	Перед определяемыми пользователем типами в модуле класса должно использоваться ключевое слово **Private**.
	Процедуры	Общая, если перед именем не использовано ключевое слово **Private**	Процедуры обработки событий в модулях форм и отчетов по умолчанию являются личными.

Примечания

- Описанные в стандартном модуле общие элементы доступны во всех процедурах текущей базы данных и баз данных, на которые имеется ссылка, если только модуль не содержит

инструкцию **Option Private Module**. Если эта инструкция присутствует, общие элементы доступны только в процедурах текущей базы данных.

- Модули класса по умолчанию являются личными. Описанные в модуле класса общие элементы доступны во всех процедурах только текущей базы данных; они не доступны в базах данных, на которые имеется ссылка. Следует отметить, что модули форм и отчетов являются модулями классов.
- Описанные в базе данных Microsoft Access переменные, константы, определяемые пользователем типы и инструкции **Declare** никогда не доступны в проектах других приложений.
- Для обращения к процедуре или переменной уровня модуля в модуле класса следует указать перед их именем имя класса. Например, если в модуле класса с именем Class1 описана переменная уровня модуля `varOrders`, к ней можно обращаться, используя синтаксис `Class1.varOrders`. Это необходимо при использовании переменной в стандартном модуле или в окне отладки. Если имя класса или формы состоит из нескольких слов, следует заключать его в скобки.
- Описанная в модуле класса переменная уровня модуля остается доступной все время существования экземпляра класса. При уничтожении экземпляра класса переменная становится недоступной
- В стандартном модуле инициализированная переменная уровня модуля сохраняет свое значение независимо от того, выполняется программа или нет, до тех пор пока ей не будет присвоено новое значение, выполнен перезапуск программы или закрыта база данных.
- Если имена процедур в модуле класса и в стандартном модуле совпадают, всегда будет вызываться процедура из стандартного модуля, если только перед ее именем явно не указано имя класса из модуля класса.

Функция Date (Microsoft Access)

Для того чтобы вставить в форму или отчет текущую дату выберите в меню **Вставка** команду **Дата и время**. В форме создается поле**,** в котором свойство **Данные (ControlSource)** определяется выражением, содержащим функцию **Date**. Вид этого выражения зависит от выбранного формата представления даты. Эта команда доступна только в режиме конструктора формы или в режиме конструктора отчета.

Допускается также использование функции **Date** в выражениях в запросах или в макросах. Во всех случаях использования функции, за исключением ее использования в программе в модуле Visual Basic, обязательными являются скобки после имени функции `Date()`.

Функция Time (Microsoft Access)

Для того чтобы вставить в форму или отчет текущее значение времени выберите в меню **Вставка** команду **Дата и время**. В форме создается поле, в котором свойство **Данные (ControlSource)** задается выражением, содержащим функцию **Time**. Вид этого выражения зависит от выбранного формата представления времени. Эта команда доступна только в режиме конструктора формы или в режиме конструктора отчета.

Допускается также использование функции **Time** в выражениях в запросах или в макросах. Во всех случаях использования функции, за исключением ее использования в программе в модуле Visual Basic, обязательными являются скобки после имени функции `Time()`.

Функция Date (Microsoft Access), пример

Функция **Date** может использоваться в вычисляемом элементе управления, путем присвоения свойству **Данные (ControlSource)** этого элемента следующего значения:

```
= Date()
```

Функция **Date** может также задавать условие отбора для запроса. Например, для таблицы «Заказы», содержащей поле «Дата размещения», можно создать запрос, возвращающий все записи, в которых значение поля «Дата размещения» попадает в диапазон с 1 апреля 1995 г. по настоящее время. Для поля «Дата размещения» в ячейке **Условие отбора** введите следующее выражение:

```
Between #4-1-95# and Date()
```

Функции DateSerial, Day, Month и Year (Microsoft Access), примеры

В следующем примере функции **DateSerial, Year**, **Month** и **Day** использованы для вычисления количества дней в заданном месяце. Параметром функции DaysInMonth может быть либо дата, либо строка.

```
Function DaysInMonth(dteInput As Date) As Integer
   Dim intDays As Integer

   ' Добавим один месяц и вычтем даты, чтобы найти разницу.
   intDays = DateSerial(Year(dteInput), Month(dteInput) + 1, Day(dteInput))
_
      - DateSerial(Year(dteInput), Month(dteInput), Day(dteInput))
   DaysInMonth = intDays
   Debug.Print intDays
End Function
```

В следующей процедуре **Sub** приведено несколько способов вызова функции DaysInMonth:

```
Sub CallDaysInMonth()
   Dim intDays As Integer
   intDays = DaysInMonth(#4/1/96#)
   intDays = DaysInMonth("4-1-96")
   intDays = DaysInMonth("April 1, 1996")
End Sub
```

Функция Now (Microsoft Access), пример

Функция **Now** может использоваться в вычисляемом элементе управления для возврата текущих системных времени и даты. Например, можно создать поле и присвоить его свойству **Данные (ControlSource)** следующее выражение:

```
= Now()
```

Функция Time (Microsoft Access), пример

Функция **Time** может использоваться в вычисляемом элементе управления для возврата текущего системного времени. Например, можно создать поле и присвоить его свойству **Данные (ControlSource)** следующее выражение:

```
= Time()
```

Функция Timer (Microsoft Access), пример

Функция **Timer** служит для определения времени выполнения операции или события в программе Microsoft Access. Для этого следует поместить вызов функции **Timer** непосредственно перед и после той операции в программе, время выполнения которой необходимо измерить. Аргументом следующей функции является имя запроса, а сама функция вычисляет время выполнения этого запроса:

```
Sub QueryTimer (strQueryName As String)
   Dim sngStart As Single, sngEnd As Single
   Dim sngElapsed As Single

   sngStart = Timer                              ' Узнаем начальное время.
   DoCmd.OpenQuery strQueryName, acNormal        ' Выполним запрос.
   sngEnd = Timer                                ' Узнаем конечное время.
   sngElapsed = Format(sngEnd - sngStart, "Fixed")   ' Время выполнения.
   MsgBox ("Выполнение запроса " & strQueryName & " заняло " & sngElapsed _
      & " секунд.")
End Sub
```

Инструкция Const (Microsoft Access)

Константа**,** определенная в стандартном модуле Microsoft Access, является личной по умолчанию. Однако константа, описанная с ключевым словом **Public**, становится доступной для других модулей в текущей базе данных. Она также становится доступной для модулей, в которых определены ссылки на текущую базу данных.

Константы, описанные в модуле класса всегда являются личными; сделать такие константы общими с помощью ключевого слова **Public** невозможно.

Инструкция Declare (Microsoft Access)

В Microsoft Access процедуры, описанные с помощью инструкции **Declare**, являются общими по умолчанию. Общая инструкция **Declare** в стандартном модуле является доступной для всех процедур в текущей базе данных и во всех базах данных, на которые имеются ссылки. Для того чтобы сделать процедуру доступной только из текущего модуля, следует описать ее с ключевым словом **Private**.

Для использования инструкции **Declare** в модуле класса предшествующее ключевое слово **Private** является обязательным. Если оно отсутствует, при компиляции Microsoft Access выдаст сообщение об ошибке.

Инструкция Dim (Microsoft Access)

При описании объектных переменных некоторых типов допускается использование ключевого слова **New** с инструкцией **Dim**. Включение ключевого слова **New** в описание переменной указывает создание нового объекта, с которым связывается объектная переменная. Таким образом, при описании объектной переменной с ключевым словом **New** нет необходимости использовать инструкцию **Set**.

С помощью ключевого слова **New** можно создать объектную переменную, связанную с объектом любого типа. Чаще всего ключевое слово **New** используется для создания экземпляра класса или нового объекта **Collection** (семейство), что показано на следующем примере:

```
' Создает объектную переменную и связывает ее с новым объектом.
Dim colInstances As New Collection
```

В следующем примере ключевое слово **New** используется для создания экземпляра класса:

```
' Создает новый экземпляр модуля класса.
Dim obj As New Class1
```

Возможно использование ключевого слова **New** для создания нового объекта Microsoft Access из некоторого компонента, поддерживающего программирование. Чтобы выяснить, поддерживает ли компонент такой синтаксис, следует обратиться к его документации.

Например, если в другом компоненте создана ссылка на библиотеку типов Microsoft Access, то новый объект Microsoft Access **Application** создается с помощью следующей инструкции:

```
Dim appAccess As New Access.Application
```

Инструкция Function (Microsoft Access)

В Microsoft Access процедура-функция **Function** становится доступной для всех остальных процедур в текущей базе данных и во всех других базах данных Microsoft Access, на которые имеются ссылки. Однако для других приложений такая процедура остается недоступной.

Если процедура **Function** с помощью ключевого слова **Private** описана в любом модуле как личная, она становится доступной только для остальных процедур из того же модуля.

Процедура **Function**, описанная как общая в личном модуле, например в модуле класса, является доступной для всех остальных процедур в этой базе данных, но недоступной из других баз данных Microsoft Access.

Инструкция Option Compare (Microsoft Access)

В Microsoft Access все модули по умолчанию содержат в разделе описаний инструкцию **Option Compare Database**. Данная инструкция указывает, что в этом модуле используется метод сравнения строк, определенный для всей базы данных.

Для изменения используемого в базе данных метода сравнения строк выберите в меню **Сервис** команду **Параметры**, выберите вкладку **Общие** и выберите нужное значение в раскрывающемся списке **Порядок сортировки базы**. По умолчанию задается «Обычный порядок», указывающий сортировку по английскому алфавиту без учета регистра.

Изменение параметра, выбираемого в поле со списком **Порядок сортировки базы данных**, не приводит к изменению способа сравнения строк в текущей базе данных. Затрагиваются толко базы данных, создаваемые после изменения этого параметра.

Для того чтобы использовать в отдельном модуле метод сравнения строк, отличный от заданного для базы данных, следует заменить инструкцию **Option Compare Database** на **Option Compare Binary** или **Option Compare Text**. Инструкция **Option Compare Binary** задает сравнение строк с помощью значений кодов ASCII, соответствующих символам, в результате чего сравнение и сортировка будут выполняться с учетом регистра. Инструкция **Option Compare Text** задает выполнение операций без учета регистра.

Примечание. Если используется свойство **Bookmark** объекта **Recordset** (набор записей), необходимо включить в раздел описаний модуля инструкцию **Option Compare Binary**. Значение свойства **Bookmark** задается и возвращается в виде массива **Variant**, содержащего данные типа **Byte**. Если задан метод сравнения строк без учета регистра, то использование свойства **Bookmark** может привести к переходу на неверную запись.

Инструкция Option Explicit (Microsoft Access)

В Microsoft Access возможно установить режим, при котором во все модули будет автоматически добавляться инструкция **Option Explicit**. Для этого следует выбрать в меню **Сервис** команду **Параметры**, выбрать вкладку **Модуль** и установить в группе **Программирование** флажок **Явное описание переменных**.

После установки этого флажка данный режим автоматически включается для всех других баз данных, создаваемых или открываемых в Microsoft Access.

Примечание. Задание данного режима приводит к включению инструкции **Option Explicit** только в новые модули. Для существующих модулей следует в явном виде ввести инструкцию **Option Explicit** в раздел описаний или вызвать при открытом модуле окно диалога **Параметры** из меню **Сервис** и установить флажок **Явное описание переменных**.

Инструкция Option Private (Microsoft Access)

При включении инструкции **Option Private** в стандартный модуль в Microsoft Access любые общие переменные и процедуры в модуле остаются доступными для всех других процедур в текущей базе данных, но становятся недоступными для процедур из других баз данных.

Поскольку по умолчанию модули класса являются личными, добавление в модуль класса инструкции **Option Private** приводит к ошибке.

Инструкция Property Get (Microsoft Access)

Инструкция **Property Get** позволяет описать процедуру обработки свойств, возвращающую значение свойства. Чаще всего инструкцию **Property Get** используют совместно с инструкцией **Property Let** для создания свойства в модуле класса. Например, можно задать новое свойство в модуле класса с помощью инструкции **Property Let** и использовать инструкцию **Property Get** для возвращения значения этого свойства. При создании нового экземпляра класса, процедуры обработки свойств ведут себя так же, как специальные свойства для нового объекта. Более подробные сведения о модулях класса можно найти в справочной системе по ключевым словам «модули класса».

Процедуры обработки свойств, описанные в модуле класса, являются личными по умолчанию и доступными для процедур из других модулей в текущей базе данных, если только при описании не использовано ключевое слово **Private**.

Инструкция Property Let (Microsoft Access)

Инструкция **Property Let** позволяет описать процедуру обработки свойств, присваивающую свойству значение. Чаще всего инструкцию **Property Let** используют совместно с инструкцией **Property Get** для создания свойства в модуле класса. Например, можно задать новое свойство в модуле класса с помощью инструкции **Property Let** и использовать инструкцию **Property Get** для возвращения значения этого свойства. При создании нового экземпляра класса, процедуры обработки свойств ведут себя так же, как специальные свойства для нового объекта. Более подробные сведения о модулях класса можно найти в справочной системе по ключевым словам «модули класса».

Процедуры обработки свойств, описанные в модуле класса, являются личными по умолчанию и доступными для процедур из других модулей в текущей базе данных, если только при описании не использовано ключевое слово **Private**.

Инструкция Public (Microsoft Access)

По умолчанию, описанные на уровне модуля переменные являются личными для данного модуля. Для того чтобы определить процедуру как общую следует в явном виде описать ее с инструкцией **Public**.

Переменные уровня модуля, описанные в стандартном модуле с инструкцией **Public,** являются доступными для всех процедур во всех модулях текущей базы данных и баз данных Microsoft Access, на которые имеется ссылка. Однако они являются недоступным для всех других приложений, кроме Microsoft Access.

Если общая переменная описана модуле, содержащем инструкцию **Option Private**, или в модуле класса, она доступна во всех процедурах текущей базы данных, но недоступна в процедурах в других базах данных.

Примечание. Не допускается использование инструкции **Public** в модуле класса для описания константы, строковой переменной фиксированной длины или массива. Кроме того, если в модуль класса включена инструкция **Declare** перед ней необходимо записать ключевое слово **Private**. Более подробные сведения о модулях класса можно найти в справочной системе по ключевым словам «Declare - инструкция».

Инструкция Sub (Microsoft Access)

В Microsoft Access общая процедура-подпрограмма **Sub**, описанная в стандартном модуле, является доступной для всех процедур во всех модулях текущей базы данных и баз данных Microsoft Access, на которые имеется ссылка. Однако она недоступна из других приложений.

Если процедура **Sub** каком-либо модуле является личной, она доступна только для процедур этого модуля.

Если процедура **Sub** описана как общая в личном модуле, например в модуле класса, она доступна во всех процедурах в данной базе данных, но не доступна в других базах данных Microsoft Access.

При создании процедуры обработки событий в форме или отчете Microsoft Access автоматически создает заготовку программы для процедуры-подпрограммы **Sub** с предшествующим ключевым словом **Private**. Например, если создать кнопку в новой форме, указать для нее [Процедура обработки событий] в свойстве **Нажатие кнопки (OnClick)** и нажать кнопку построителя  для перехода в окно модуля формы, то в модуле появятся следующие инструкции:

```
Private Sub Кнопка0_Click

End Sub
```

Пользователь должен ввести текст программы, которая будет выполняться при возникновении события **Нажатие кнопки (Click)** для этой кнопки.

Инструкция Type (Microsoft Access)

В Microsoft Access типы, определяемые пользователем, являются общими по умолчанию. В модуле класса невозможно описать определяемый пользователем общий тип. Чтобы описать такой тип, следует использовать ключевое слово **Private** перед инструкцией **Type**. Если ключевое слово **Private** пропущено, при компиляции Microsoft Access выдаст сообщение об ошибке.

Инструкция Const (Microsoft Access), пример

В Microsoft Access в разделах описаний модуля класса не допускается описание общих констант. Константы в модуле класса должны быть личными. Однако возможно описание общих и личных констант в области описаний стандартного модуля.

```
' В модуле класса.
Const conCommission As Single = .08

' В стандартном модуле.
Const conErrorNumber As Integer = 91
Public Const conAddress As String = "123456 Москва, Тверская 1313"
```

Инструкция Declare (Microsoft Access), пример

В Microsoft Access инструкция **Declare** позволяет описать на уровне модуля в стандартном модуле ссылку на внешнюю процедуру из библиотеки динамической компоновки (DLL). Инструкции **Declare** являются общими по умолчанию. Допускается также использование инструкции **Declare** с предшествующим ключевым словом **Private** в модуле класса.

В следующем примере создается ссылка на процедуру из библиотеки DLL, после чего процедура вызывается в программе Visual Basic.

```
' В стандартном модуле.
Declare Sub MessageBeep Lib "User32" (ByVal intN As Integer)

' В модуле класса.
Private Declare Sub MessageBeep Lib "User32" (ByVal intN As Integer)

' После создания ссылки на процедуру из библиотеки DLL в инструкции
' Declare эта процедура вызывается в программе обычным образом.
Sub SystemBeep(intBeeps As Integer, sngPause As Single)
   Dim intI As Integer
   Dim sngStart As Single

   For intI = 1 To intBeeps
      ' Вызов системной функции.
      Call MessageBeep(intBeeps)
      ' Начало отсчета времени.
      sngStart = Timer
      ' Пауза между сигналами.
      Do While Timer < sngStart + sngPause
         ' Возврат управления операционной системе.
         DoEvents
      Loop
   Next intI
End Sub
```

Неверное использование ключевого слова New. (Microsoft Access)

В Microsoft Access модули форм и модули отчетов являются модулями класса. Ключевое слово **New** позволяет создать новый экземпляр модуля класса и связать его с объектной переменной.

В следующем примере демонстрируется создание с помощью ключевого слова **New** нового экземпляра модуля класса формы с именем Form_Заказы.

```
Dim frmOrdersObject As New Form_Заказы
```

Не допускается использование констант, строк постоянной длины, массивов и инструкций Declare в качестве общих компонентов модуля класса. (Microsoft Access)

В Microsoft Access модули класса могут существовать самостоятельно или быть связанными с формами и отчетами. В модуле формы или отчета не допускается описание общих констант и строк фиксированной длины или использование инструкции **Declare**. Необходимо помещать перед ними ключевое слово **Private**.

Компонент уже существует в данной форме. (Microsoft Access)

В Microsoft Access не допускается использование в модуле формы или модуле отчета идентификатора, имя которого совпадает с именем метода или свойства формы или отчета. Например, форма имеет свойство **RecordSource**, поэтому включение следующей процедуры в модуль формы приведет к ошибке.

```
Sub RecordSource()    ' Вызывает ошибку.
    .                    ' Текст программы.
    .
    .
End Sub
```

Функция DoEvents (Microsoft Access)

В Microsoft Access вызов функции **DoEvents** игнорируется в следующих случаях:

- в определяемой пользователем функции или процедуре**,** рассчитывающей значение поля в запросе, форме или отчете;
- в определяемой пользователем функции, создающей список в поле со списком, объект-список или объект OLE.

При вызове функции **DoEvents** в любой из этих ситуаций Microsoft Access не передает управление операционной системе.

Функция IsDate (Microsoft Access), пример

В данном примере выводится приглашение ввести строковое значение, функция **IsDate** проверяет возможность преобразования строкового значения в дату, после чего выводится соответствующее сообщение.

```
Sub CheckDate()
   Dim strDate As String

   strDate = InputBox("Введите строковое значение для преобразования в
дату.")
   ' Проверка переменной.
   If IsDate(strDate) Then
      ' Если строка представляет дату, форматирует ее и выводит в
диалоговом окне.
      MsgBox "Дата: " & Format(DateValue(strDate), "Long Date")
   Else
      MsgBox "Введенное значение не является датой."
   End If
End Sub
```

Функция IsNull (Microsoft Access), пример

В данном примере функция **IsNull** проверяет, имеет ли элемент управления пустое (**Null**) значение. Если да, выводится приглашение ввести данные. Если элемент управления имеет присвоенное значение, выводится сообщение с этим значением.

```
Sub ControlValue(ctlText As Control)
   Dim strMsg As String

   ' Проверяет, что элемент управления является полем.
   If ctlText.ControlType = acTextBox Then
      ' При значении Null выводит приглашение ввести данные.
      If IsNull(ctlText.Value) Then
         strMsg = "Пустое поле '" & ctlText.Name & "'." _
         & vbCrLf & "Пожалуйста введите значение данного поля."
            If MsgBox(strMsg, vbQuestion) = vbOK Then
               Exit Sub
            End If
      ' Если поле имеет непустое значение, выводит это значение.
      Else
         MsgBox (ctlText.Value)
      End If
   End If
End Sub
```

Функция IsObject (Microsoft Access), пример

В данном примере функция **IsObject** проверяет, является ли аргумент, переданный в функцию, объектом.

```
Sub CheckObject(varAny As Variant)
   Dim varX as Variant

   If IsObject(varAny) Then
      Set varX = varAny
   Else
      varX = varAny
   End If
End Sub
```

Функция TypeName (Microsoft Access), пример

В данном примере несколько объектных переменных создаются и передаются в функцию **TypeName**.

```
Sub ObjectTypes()
   Dim dbs As Database, tdf As TableDef
   Dim fld As Field

   ' Связывает переменную с текущей базой данных.
   Set dbs = CurrentDb
   ' Возвращает объект TableDef, связанный с таблицей «Заказы».
   Set tdf = dbs.TableDefs("Заказы")
   ' Возвращает объект Field, связанный с полем «ДатаРазмещения».
   Set fld = tdf.Fields("ДатаРазмещения")
   ' Для каждого объекта печатает значение, возвращаемое функцией TypeName.
   Debug.Print TypeName(dbs)
   Debug.Print TypeName(tdf)
   Debug.Print TypeName(fld)
End Sub
```

Константы компилятора (Microsoft Access)

Microsoft Access является 32-разрядным приложением. Константа компилятора **Win32** всегда имеет значение **True** (-1)**,** тогда как константа **Win16** всегда имеет значение **False** (0).

Функция MsgBox (Microsoft Access)

Функция **MsgBox** используется в Microsoft Access для создания форматированных сообщений об ошибках, аналогичных встроенным сообщениям об ошибках, выводящимся в Microsoft Access. Данная функция позволяет указать в первом аргументе *текст* строку, содержащую три части, разделяемые символом (@).

В следующем примере демонстрируется вывод форматированного диалогового окна со строкой сообщения, разбитой на три части. Первая часть строки выводится как заголовок полужирным шрифтом. Вторая часть образует основной текст сообщения под этим заголовком. Третья часть выводится обычным шрифтом под заголовком «Решение», который автоматически создается функцией.

```
MsgBox "Неверная кнопка!@Данная кнопка не работает.@Нажмите другую.", _
vbOKOnly + vbExclamation
```

Примечание. Для вывода в Microsoft Access окна сообщения без значка опустите константу, определяющую значок, или укажите для второго аргумента *тип*, определяющего выводящиеся в окне сообщения значок и кнопки, значение 0, как в следуем примере.

```
MsgBox "Сообщение без значка.", 0
```

Функция InputBox (Microsoft Access)

В Microsoft Access не допускается открытие файла справки из диалогового окна, создаваемого функцией **InputBox**. Указанные значения аргументов *файлСправки* и *контекст* будут проигнорированы, а кнопка **Справка** отображена не будет. Однако Microsoft Access при этом не генерирует ошибку.

Функция MsgBox (Microsoft Access), пример

В следующем примере функция **MsgBox** используется для создания в Microsoft Access форматированного сообщения об ошибке. Обратите внимание на символы (@), разделяющие части сообщения.

```
Sub CustomMessage()
   Dim strMsg As String, strInput As String

   ' Инициализирует строку.
   strMsg = " Число вне диапазона.@Введено число, меньшее 1 " _
      & " или превышающее 10.@Нажмите кнопку 'OK' для повторения " _
      & " ввода числа."
   ' Приглашение пользователю ввести число.
   strInput = InputBox("Введите число в диапазоне от 1 до 10.")
   ' Проверка, не нажата ли кнопка «Отмена».
   If strInput <> "" Then
      ' Проверка введенного пользователем значения.
      Do While strInput < 0 Or strInput > 10
         If MsgBox(strMsg, vbOKCancel, "Ошибка!") = vbOK Then
            strInput = InputBox("Введите число в диапазоне от 1 до 10.")
         Else
            Exit Sub
         End If
      Loop
      ' Выводит допустимое число, введенное пользователем.
      MsgBox "Введено число " & strInput & "."
   Else
      Exit Sub
   End If
End Sub
```

Функция InputBox (Microsoft Access), пример

В следующем примере функция **InputBox** используется для возвращения имени пользователя. Обратите внимание, что в Microsoft Access не допускается открытие файла справки из диалогового окна, создаваемого функцией **InputBox**.

```
Sub Greeting()
   Dim strInput As String, strMsg As String

   strMsg = "Введите ваше имя."
   strInput = InputBox(Prompt:=strMsg, Title:="Пользователь", XPos:=2000,
YPos:=2000)
   MsgBox "Привет, " & strInput
End Sub
```

Обзор панелей команд (Microsoft Access)

Для создания в Microsoft Access 97 программ, включающих панели команд, необходимо сначала определить ссылку на объектную библиотеку Microsoft Office. Для этого, находясь в режиме конструктора модуля, из меню **Сервис** следует выбрать команду **Ссылки** и установить флажок **Microsoft Office 8.0 Object Library**.

После установки ссылки на объектную библиотеку Microsoft Office объекты, свойства и методы Microsoft Office отображаются в окне просмотра объектов. Для программирования панелей команд используются объекты **CommandBar**, **CommandBarControl**, **CommandBarButton**, **CommandBarComboBox** и **CommandBarPopup**, а также семейства **CommandBars** и **CommandBarControls**.

Использование панелей команд (Microsoft Access)

В Microsoft Access 97 для изменения панелей команд во время разработки служит диалоговое окно **Настройка**, вызываемое по команде **Панели инструментов** из меню **Вид** и выбору пункта **Настройка.** Это диалоговое окно используется для добавления или изменения строк меню, панелей инструментов и контекстных меню, а также для добавления любых встроенных элементов меню или кнопок панелей инструментов в эти панели команд. Однако в этом диалоговом окне возможно добавление только специальных кнопок и элементов меню в панели команд. Для добавления в панель команд специальных полей редактирования, раскрывающихся списков или полей со списком следует использовать Visual Basic.

Использование модели объекта Панель команд в Microsoft Access

Для создания в Microsoft Access 97 программ, включающих панели команд, необходимо сначала определить ссылку на объектную библиотеку Microsoft Office. Для этого, находясь в режиме конструктора модуля, из меню **Сервис** следует выбрать команду **Ссылки** и установить флажок **Microsoft Office 8.0 Object Library**.

Добавление и отображение контекстных меню (Microsoft Access)

Microsoft Access 97 предоставляет интерфейс пользователя для добавления и изменения контекстных меню. Чтобы добавить специальное контекстное меню, следует сначала в диалоговом окне **Настройка**, вызываемому по команде **Панели инструментов** из меню **Вид** и выбору пункта **Настройка**, добавить панель инструментов, а затем в диалоговом окне **Свойства панели инструментов** изменить **Тип** панели инструментов на **Всплывающее окно**. Для удаления контекстного меню достаточно изменить его тип на **Строка меню** или **Панель инструментов**, а затем удалить. В Visual Basic допускается добавление, изменение и удаление любых контекстных меню.

Встроенные и специальные контекстные меню отображаются в панели инструментов **Контекстные меню**, появляющейся при установке флажка **Контекстные меню** в диалоговом окне **Настройка**. С помощью этой панели инструментов выполняется добавление кнопок и элементов меню в контекстные меню, а также их удаление из меню. Однако следует отметить, что для обращения к контекстному меню из Visual Basic используется синтаксис **CommandBars("***имяконтекстногоменю***")**. Команда `CommandBars("Контекстные меню")` бессмысленна.

Для отображения контекстного меню в ответ на нажатие правой кнопки мыши следует воспользоваться процедурой обработки события.. В следующем примере при нажатии правой кнопки мыши в пустой части области выделения записи отображается специальное контекстное меню "КонтекстноеМенюФормы".

```
Private Sub Form_КнопкаВниз(Button As Integer, Shift As Integer, _
      X As Single, Y As Single)
   Dim cmdBar As CommandBar
   Dim intRight As Integer

   Set cmdBar = CommandBars("КонтекстноеМенюФормы")
   intRight = (Button and acRightButton) > 0
      If intRight Then
         cmdBar.ShowPopup 200, 200
      End If
End Sub
```

Объект семейство CommandBars (Microsoft Access)

Хотя в большинстве семейств Access отсчет элементов начинается с нуля, во всех семействах панелей команд отсчет начинается с 1. Следует заметить однако, что для перебора всех объектов коллекции можно использовать синтаксис **For Each...Next**, не заботясь об индексации.

Объект семейство CommandBarControls (Microsoft Access)

Хотя в большинстве семейств Access отсчет элементов начинается с нуля, во всех семействах панелей команд отсчет начинается с 1. Следует заметить однако, что для перебора всех объектов коллекции можно использовать синтаксис **For Each...Next**, не заботясь об индексации.

Свойство «Имя» (Name) - (Microsoft Access)

В Microsoft Access 97 с помощью диалогового окна **Свойства панели инструментов** можно изменять имена специальных строки меню, панели инструментов, или контекстного меню. Для этого в меню **Вид** следует выбрать пункт **Панели инструментов**, а затем **Настройка**, и в диалоговом окне **Настройка** нажать кнопку **Свойства**. Затем в поле **Выбранная панель** следует выбрать нужные специальную строку меню, панель инструментов или контекстное меню, и в поле **Название** ввести новое имя. Переименовать встроенные строку меню, панель инструментом или контекстное меню невозможно.

Свойство Type (Microsoft Access)

В Microsoft Access 97 с помощью диалогового окна **Свойства панели инструментов** можно изменять тип специальных строки меню, панели инструментов, или контекстного меню. Для этого в меню **Вид** следует выбрать пункт **Панели инструментов**, а затем **Настройка**, и в диалоговом окне **Настройка** нажать кнопку **Свойства**. Затем в поле **Выбранная панель** следует выбрать нужные специальную строку меню, панель инструментов или контекстное меню, и в поле **Тип** задать нужный тип. Изменить тип встроенных строки меню, панели инструментов или контекстного меню невозможно.

Если в качестве типа панели команд выбрана строка Всплывающее окно, эта панель исчезает из списка **Панели инструментов** диалогового окна **Настройка** и перемещается в строку меню, отображаемую при установке флажка **Контекстные меню** в списке **Панели инструментов** (в этой строке меню выводятся все контекстные меню Microsoft Access).

Свойство Protection (Microsoft Access)

В Microsoft Access 97 с помощью диалогового окна **Свойства панели инструментов** можно задавать различные параметры для свойства **Protection**, запрещающего или позволяющего настройку пользователем строки меню, панели инструментов или специального контекстного меню. Для этого в меню **Вид** следует выбрать пункт **Панели инструментов**, а затем **Настройка**, и в диалоговом окне **Настройка** нажать кнопку **Свойства**. Затем в поле **Выбранная панель** следует выбрать нужные специальную строку меню, панель инструментов или контекстное меню, и установить требуемую защиту с помощью поля со списком **Закрепление**, флажков **Настройка**, **Изменение размера**, **Перемещение** и **Отображение/скрытие**. Для специальных контекстных меню доступен только параметр **Настройка**.

Метод Reset (Microsoft Access)

В Microsoft Access 97 с помощью диалогового окна **Свойства панели инструментов** можно выполнить сброс встроенных строки меню или панели инструментов. Для этого в меню **Вид** следует выбрать пункт **Панели инструментов**, а затем **Настройка**, и в диалоговом окне **Настройка** нажать кнопку **Свойства**. Затем в поле **Выбранная панель** следует выбрать нужные специальную строку меню или панель инструментов и нажать кнопку **Сброс**. Невозможно сбросить специальные строки меню, а также определенные контекстные меню.

Кнопка **Сброс** в диалоговом окне **Настройка** сбрасывает выбранные встроенную строку меню или панель инструментов, а также и встроенные контекстные меню, если в диалоговом окне **Настройка** установлен флажок **Контекстные меню**. Команда **Сброс** в контекстном меню, появляющемся при щелчке правой кнопкой мыши определенного встроенного элемента управления в строке меню или панели инструментов в диалоговом окне **Настройка**, возвращает этот элемент управления в исходное состояние. Кнопка и команда **Сброс** выполняют те же действия, что и вызов метода **Reset**.

Кнопка **Сброс** сбрасывает не только закрепленные за строкой меню или панелью инструментов текст, картинку и выполняемую команду, но и восстанавливают исходные параметры строки меню или панели инструментов, включая исходный размер, положение и видимость.

Свойство «Подпись» (Caption) - (Microsoft Access)

В Microsoft Access 97 диалоговое окно **Свойства элемента <имя>** панели команд позволяет задать подпись для элемента панели команд. Для его вызова следует в меню **Вид** выбрать команду **Панели инструментов** и подкоманду **Настройка**. Для строки меню и панели инструментов необходимо отобразить их, а затем щелкнуть правой кнопкой мыши элемент, для которого задается свойство **Подпись (Caption)**. Для элементов контекстного меню следует на вкладке **Панели инструментов** диалогового окна **Настройка** установить флажок **Контекстные меню**, затем в появившейся строке меню найти требуемый элемент и щелкнуть его правой кнопкой мыши. Далее следует выбрать команду **Свойства** и в поле **Подпись** ввести необходимое значение.

При отображении свойства элемента управления **Подпись (Caption)** в диалоговом окне **Свойства элемента <имя>** или в Visual Basic, перед клавишей быстрого вызова помещается амперсанд (&) (например, Вст&авить). При обращении к элементу управления из Visual Basic можно использовать или не использовать амперсанд. Например, следующие ссылки идентичны:

```
CommandBars("Строка меню").Controls("Сервис")
```

```
CommandBars("Строка меню").Controls("&Сервис")
```

При задании значения свойства **Подпись (Caption)** использование символа амперсанд необходимо, если элемент управления должен иметь клавишу быстрого вызова.

Свойство ShortcutText (Microsoft Access)

В Microsoft Access 97 диалоговое окно **Свойства элемента <имя>** панели команд служит для установки сочетания клавиш, отображаемого справа от встроенной кнопки в меню, подменю или контекстном меню. Для его вызова в меню **Вид** следует выбрать пункт **Панели инструментов**, а затем **Настройка**. Для строки меню и панели инструментов необходимо отобразить их, а затем щелкнуть правой кнопкой мыши элемент, чье свойство **ShortcutText** требуется установить. Для элементов контекстного меню следует на вкладке **Панели инструментов** диалогового окна **Настройка** установить флажок **Контекстные меню**, затем в появившейся строке меню найти требуемый элемент и щелкнуть его правой кнопкой мыши. Далее следует выбрать команду **Свойства** и в поле **Текст ярлыка** ввести нужное сочетание клавиш. Значение этого свойства в Microsoft Access может быть задано для любой встроенной кнопки, но не для специальной кнопки.

Установка в Microsoft Access этого свойства приводит к отображению указанного сочетания клавиш рядом с элементом управления. При изменении или добавлении сочетания клавиш для встроенного элемента управления, включенная в него клавиша или сочетание клавиш быстрого вызова не приведет к запуску команды, закрепленной за этим элементом. Для запуска команды необходимо назначить клавишу или сочетание клавиш с помощью макроса AutoKeys. В этом макросе клавише или сочетании клавиш следует назначить макрос, использующий макрокоманду **ВыполнитьКоманду (RunCommand)** для выполнения команды.

Для специальных команд рядом с элементом управления нельзя отобразить сочетание клавиш, но можно с помощью макроса AutoKeys назначить клавишу или сочетание клавиш, которые будут выполнять тот же набор действий, что и макрос или функция, указанная в свойстве **OnAction** элемента управления. Необходимо либо закрепить клавишу или сочетание клавиш за определенным набором макрокоманд, либо воспользоваться макрокомандами **ЗапускМакроса (RunMacro)** или **ЗапускПрограммы (RunCode)** для запуска требуемого макроса или функции.

Свойство TooltipText (Microsoft Access)

В Microsoft Access 97 диалоговое окно **Свойства элемента <имя>** панели команд служит для установки всплывающей подсказки для элемента управления на панели команд. Для его вызова в меню **Вид** следует выбрать пункт **Панели инструментов**, а затем **Настройка**. Для строки меню и панели инструментов необходимо отобразить их, а затем щелкнуть правой кнопкой мыши элемент, чье свойство **TooltipText** требуется задать. Для элементов контекстного меню следует на вкладке **Панели инструментов** диалогового окна **Настройка** установить флажок **Контекстные меню**, затем в появившейся строке меню найти требуемый элемент и щелкнуть его правой кнопкой мыши. Далее следует выбрать команду **Свойства** и в поле **Всплывающая подсказка** ввести нужный текст.

Свойство OnAction (Microsoft Access)

В Microsoft Access 97 диалоговое окно **Свойства элемента <имя>** панели команд служит для задания значения свойства **OnAction** элемента управления на панели команд. Для его вызова в меню **Вид** следует выбрать пункт **Панели инструментов**, а затем **Настройка**. Для строки меню и панели инструментов необходимо отобразить их, а затем щелкнуть правой кнопкой мыши элемент, чье свойство **OnAction** требуется задать. Для элементов контекстного меню следует на вкладке **Панели инструментов** диалогового окна **Настройка** установить флажок **Контекстные меню**, затем в появившейся строке меню найти требуемый элемент и щелкнуть его правой кнопкой мыши. Далее следует выбрать команду **Свойства** и в поле **Действие** ввести имя запускаемого макроса или функции.

Свойство **OnAction** может использоваться только для запуска макроса или функции. Для запуска макроса выберите его из раскрывающегося списка в поле **Действие** или введите там его имя. Для запуска функции введите ее имя в виде **=***имяфункции***( )**.

Значением свойства **OnAction** может быть также выражение. Оно может включать встроенные или определяемые пользователем функции, а также сочетания функций. При вычислении выражения Microsoft Access выполняет все операции, входящие в это выражение. Например, следующее выражение запускает определяемую пользователем функцию "СпециальнаяФункция":

```
=СпециальнаяФункция()
```

В Visual Basic значением свойства **OnAction** должна быть строка, поэтому для задания выражения необходимо передавать его в строковом виде. При передачи строкового значения следует обращать особое внимание на заключение строки внутри другой строки в кавычки ("). (Следует отметить, что в поле **Действие** диалогового окна **Свойства элемента <имя>** строка в выражении должна быть заключена лишь в один набор кавычек.) Также необходимо следить за наличием знака равенства (=), указывающего, что строка содержит выражение. Если функция не имеет аргументов, в строку надо обязательно включить скобки [( )]. Следующая строка, присвоенная свойству **OnAction**, аналогична приведенному выше примеру:

```
"=СпециальнаяФункция()"
```

Следует отметить, что значением свойства **OnAction** может быть имя только процедуры **Function**; нельзя присвоить ему имя процедуры **Sub.**.

Для элементов специальных панелей команд значение этого свойства должно быть задано, чтобы элемент управления выполнял какие-либо действия. При определении этого свойства для встроенного элемента управления новое значение свойства **OnAction** замещает обычную команду, выполняемую элементом; т. е. можно добиться выполнения встроенным элементом управления функций, отличающихся от заданных по умолчанию. Такой элемент становится по существу специальным.

Свойство Style (Microsoft Access)

В Microsoft Access 97 диалоговое окно **Свойства элемента <имя>** панели команд служит для задания стиля отображения кнопок на панели команд. Для его вызова в меню **Вид** следует выбрать пункт **Панели инструментов**, а затем **Настройка**. Для строки меню и панели инструментов необходимо отобразить их, а затем щелкнуть правой кнопкой мыши элемент, чье свойство **Style** требуется установить. Для элементов контекстного меню следует на вкладке **Панели инструментов** диалогового окна **Настройка** установить флажок **Контекстные меню**, затем в появившейся строке меню найти требуемый элемент и щелкнуть его правой кнопкой мыши. Далее следует выбрать команду **Свойства** и в поле со списком **Стиль** выбрать необходимый стиль.

Значения в поле **Стиль** соответствуют встроенным константам, используемым для задания значения свойства **Style**.

Значение в поле «Стиль»	**Константа свойства Style**
По умолчанию	**msoButtonAutomatic**
Только текст (всегда)	**msoButtonCaption**
Только текст (в меню)	**msoButtonIcon**
Рисунок и текст	**msoButtonIconAndCaption**
Без надписи	**msoComboNormal**
Показать надпись	**msoComboLabel**

Свойство «Файл справки» (HelpFile) - (Microsoft Access)

В Microsoft Access 97 диалоговое окно **Свойства элемента <имя>** панели команд служит для установки значения свойства **Файл справки (HelpFile)** для элемента управления на панели команд. Для его вызова в меню **Вид** следует выбрать пункт **Панели инструментов**, а затем **Настройка**. Для строки меню и панели инструментов необходимо отобразить их, а затем щелкнуть правой кнопкой мыши элемент, чье свойство **Файл справки (HelpFile)** требуется задать. Для элементов контекстного меню следует на вкладке **Панели инструментов** диалогового окна **Настройка** установить флажок **Контекстные меню**, затем в появившейся строке меню найти требуемый элемент и щелкнуть его правой кнопкой мыши. Далее следует выбрать команду **Свойства** и в поле **Файл справки** ввести имя нужного файла справки.

Свойство HelpContextId (Microsoft Access)

В Microsoft Access 97 диалоговое окно **Свойства элемента <имя>** панели команд служит для задания значения свойства **HelpContextId** элемента управления на панели команд. Для его вызова в меню **Вид** следует выбрать пункт **Панели инструментов**, а затем **Настройка**. Для строки меню и панели инструментов необходимо отобразить их, а затем щелкнуть правой кнопкой мыши элемент, чье свойство **HelpContextId** требуется задать. Для элементов контекстного меню следует на вкладке **Панели инструментов** диалогового окна **Настройка** установить флажок **Контекстные меню**, затем в появившейся строке меню найти требуемый элемент и щелкнуть его правой кнопкой мыши. Далее следует выбрать команду **Свойства** и в поле **Идентификатор справки** ввести номер нужного раздела справки.

Свойство Parameter (Microsoft Access)

В Microsoft Access 97 диалоговое окно **Свойства элемента <имя>** панели команд служит для задания значения свойства **Parameter** элемента управления на панели команд. Для его вызова в меню **Вид** следует выбрать пункт **Панели инструментов**, а затем **Настройка**. Для строки меню и панели инструментов необходимо отобразить их, а затем щелкнуть правой кнопкой мыши элемент, чье свойство **Parameter** требуется задать. Для элементов контекстного меню следует на вкладке **Панели инструментов** диалогового окна **Настройка** установить флажок **Контекстные меню**, затем в появившейся строке меню найти требуемый элемент и щелкнуть его правой кнопкой мыши. Далее следует выбрать команду **Свойства** и в поле **Параметр** ввести сведения о параметре.

Свойство «Дополнительные сведения» (Tag) - (Microsoft Access)

В Microsoft Access 97 диалоговое окно **Свойства элемента <имя>** панели команд служит для задания значения свойства **Дополнительные сведения (Tag)** элемента управления на панели команд. Для его вызова в меню **Вид** следует выбрать пункт **Панели инструментов**, а затем **Настройка**. Для строки меню и панели инструментов необходимо отобразить их, а затем щелкнуть правой кнопкой мыши элемент, чье свойство **Дополнительные сведения (Tag)** требуется задать. Для элементов контекстного меню следует на вкладке **Панели инструментов** диалогового окна **Настройка** установить флажок **Контекстные меню**, затем в появившейся строке меню найти требуемый элемент и щелкнуть его правой кнопкой мыши. Далее следует выбрать команду **Свойства** и в поле **Дополнительные сведения** ввести необходимые сведения.

Свойство BeginGroup (Microsoft Access)

В Microsoft Access 97 диалоговое окно **Свойства элемента <имя>** панели команд служит для определения находится ли элемент управления панели команд в начале группы элементов. Перед элементом управления в панели команд отображается строка-разделитель. Для вызова этого диалогового окна в меню **Вид** следует выбрать пункт **Панели инструментов**, а затем **Настройка**. Для строки меню и панели инструментов необходимо отобразить их, а затем щелкнуть правой кнопкой мыши элемент, чье свойство **BeginGroup** требуется задать. Для элементов контекстного меню следует на вкладке **Панели инструментов** диалогового окна **Настройка** установить флажок **Контекстные меню**, затем в появившейся строке меню найти требуемый элемент и щелкнуть его правой кнопкой мыши. Далее следует выбрать команду **Свойства** и в установить или снять флажок **Создать группу**.

Разделитель, отображаемый перед началом группы элементов управления, не является элементом управления и не влияет на число элементов и индексы семейства **CommandBarControls**.

Метод Add (Семейство CommandBarControls) (Microsoft Access)

Аргументом ***Id*** этого метода должно быть значение типа **Integer,** соответствующее встроенной кнопке или элементу меню. Это приведет к тому, что новый элемент управления будет иметь внешний вид и функциональность, схожие с существующими кнопкой или элементом меню.

Однако если аргументом ***Id*** будет значение типа **Integer** команды или элемента меню Microsoft Office, измененных Microsoft Access (например, в Microsoft Access используется другая метка, сочетание клавиш или текст вместо значка), эти изменения не будут отображены. Чтобы добавляемый панель команд элемент управления выглядел и функционировал точно также, как соответствующий элемент Microsoft Access, следует воспользоваться методом **Copy** для копирования элемента управления на панель команд.

Свойство Id (Microsoft Access)

Значение свойства **Id** может быть задано только при создании нового элемента управления. Аргументом ***Id*** метода **Add** семейства **CommandBarControls** должно быть значение типа **Integer,** соответствующее встроенной кнопке, элементу меню или одной из эквивалентных встроенных констант, используемых в методе Microsoft Access **RunCommand**. Для специальных элементов управление аргумент ***Id*** должен иметь значение 1 или быть пустым.

Свойство **Id** может быть использовано для определения значения типа **Integer,** соответствующего встроенному элементу управления. Например, чтобы определить значения типа **Integer,** соответствующее команде **Вставить** из меню **Правка** главной строки меню, следует воспользоваться следующей ссылкой:

```
CommandBars("Строка меню").Controls("Правка").Controls("Вставить").Id
```

Свойство «Доступ» (Enabled) - (Microsoft Access)

Это свойство позволяет сделать недоступным (вывести с пониженной яркостью) или доступным элемент меню или кнопку. Это свойство применимо к именам меню, именам подменю, а также именам элементов подменю. Свойство **Доступ (Enabled)** заменяет собой макрокоманду **ЗадатьКомандуМеню (SetMenuItem)** и метод **SetMenuItem** Microsoft Access 95.

Данное свойство позволяет сделать недоступным элемент меню, обычно доступный в Microsoft Access, однако нельзя сделать доступным элемент меню, обычно недоступный в Microsoft Access.

Свойство State (Microsoft Access)

Свойство **State** служит для установки или снятия пометки с элемента меню или кнопки. Это свойство заменяет собой макрокоманду ЗадатьКомандуМеню (SetMenuItem) и метод **SetMenuItem** Microsoft Access 95.

Свойство Position (Microsoft Access)

Значением свойства **Position** не может быть встроенная константа **msoBarPopup**. Для создания контекстного меню следует задать аргумент ***Position*** при вызове метода **Add** семейства **CommandBars**.

Константа **msoBarMenuBar** используется только на Macintosh.

Копирование панели команд (Microsoft Access), пример

В следующем примере панель команд и ее содержимое копируются в новую панель команд. В процедуре ВводНовогоИмени пользователю предлагается ввести имена копируемой и новой панелей команд. Эта процедура вызывает функцию КопированиеПанелиКоманд, использующую введенные имена для создания новой панели команд.

```
Sub ВводНовогоИмени()
   КопированиеПанелиКоманд "Строка меню", "Новая строка меню"
End Sub

Function КопированиеПанелиКоманд(strOrigCBName As String, strNewCBName As
String) As Boolean
   ' Эта процедура копирует панель команд с именем, заданным в переменной
strOrigCBName
   ' в новую панель команд с именем, заданным в переменной strNewCBName.
   Dim cbrOriginal As CommandBar, intOrigCount As Integer
   Dim cbrCopy As CommandBar, cbrCtl As CommandBarControl

   On Error GoTo CBCopy_Err

   Set cbrOriginal = CommandBars(strOrigCBName)
   intOrigCount = cbrOriginal.Controls.Count
   Set cbrCopy = CommandBars.Add(strNewCBName)
   ' Вывод новой панели команд на экран.
   cbrCopy.Visible = True

   For Each cbrCtl In cbrOriginal.Controls
      cbrCtl.Copy cbrCopy
   Next cbrCtl
Exit Function

CBCopy_Err:
   MsgBox Err.Description
   Exit Function
End Function
```

Создание специальной кнопки (Microsoft Access), пример

В следующем примере в заданной панели команд создается специальная кнопка. В процедуре ЗаданиеПанелиКомандИКнопки пользователю предлагается ввести имя панели команд, на которую требуется добавить кнопку, и имя кнопки. Эта процедура вызывает функцию СозданиеСпециальнойКнопки, использующую введенные имена для добавления на панель команд специальной кнопки.

```
Sub ЗаданиеПанелиКомандИКнопки()
   СозданиеСпециальнойКнопки "Строка меню", "Специальная кнопка"
End Sub

Function СозданиеСпециальнойКнопки(strBarName As String, strName As String)
As Integer
   Dim cmdBar As CommandBar, cmdBarCustomButton As CommandBarButton

   Set cmdBar = CommandBars(strBarName)
   ' Если панель команд не является видимой, она выводится на экран.
   If cmdBar.Visible = False Then cmdBar.Visible = True

   On Error Resume Next

   Set cmdBarCustomButton = cmdBar.Controls(strName)
   If Err <> 0 Then
      ' Создание специальной кнопки.
      Set cmdBarCustomButton = cmdBar.Controls.Add(msoControlButton, , , ,
True)
   End If
   With cmdBarCustomButton
      .Caption = strName
      .BeginGroup = True
      .OnAction = "=msgbox(""Привет всем!"")"  ' Запуск макроса или
специальной функции.
      .Style = msoButtonCaption
   End With
End Function
```

Метод Print (Microsoft Access)

Метод **Print** из Visual Basic применяется только к объекту **Debug** Microsoft Access, необходимому для вызова данного метода. Его использование с объектом **Form** недопустимо.

Необходимо отличать метод **Print** из Visual Basic от метода **Print** из Microsoft Access, применимого только к объекту **Report**.

Метод Raise (Microsoft Access)

Метод **Raise** используется для создания ошибок Visual Basic. Все свойства объекта **Err** устанавливаются так, будто произошла ошибка выполнения. Однако данный метод применим только для работы с ошибками Visual Basic. Для ошибок Microsoft Access или ядра базы данных Microsoft метод **Raise** неприменим.

При попытке вызова метода **Raise** с номером ошибки, зарезервированным Microsoft Access или ядром базы данных, значению свойства **Description** объекта **Err** будет присвоено значение «Ошибка, определенная приложением или объектом». Для ввода строки описания ошибки Microsoft Access или ядра базы данных, не возникшей во время выполнения, используется метод **AccessError**.

Метод **AccessError** позволяет получить описание ошибки, номер которой передан ему в качестве аргумента типа **Long**.

Функция Command (Microsoft Access)

Для того чтобы изменить аргумент командной строки после открытия базы данных, выберите в меню **Сервис** команду **Параметры**. На вкладке **Другие** введите новое значение аргумента в поле **Аргументы командной строки**. После этого функция **Command** будет возвращать новое значение аргумента.

При вызове функции **Command** в любом месте, кроме программы в модуле Visual Basic, следует помещать пустые скобки после имени функции. Например, при вызове функции **Command** из поля в форме, следует ввести в ячейку свойства **Данные** (**ControlSource**) такое выражение:

```
=Command()
```

Разные форматы для разных строковых значений (функция Format) (Microsoft Access)

В версиях 1.*x* и 2.0 Microsoft Access функция **Format** использовалась для возвращения значения пустой строки и значения **Null**. Например, было возможно использование в программе с данной функцией следующего форматирующего выражения для получения соответствующего строкового значения:

```
Dim varX As Variant, varStrX As Variant
' Присвоение переменной varStrX некоторого значения и его передача в
функцию Format.
varX = Format(varStrX, "@;ZLS;Null")
```

В Microsoft Access 97 необходимо отдельное тестирование случая **Null** с возвращением соответствующего значения в зависимости от результата. Например, в выражении с функцией **Format** возможно следующее использование функции **IIf**:

```
varX = IIf(IsNull(varStrX),"Null", Format(varStrX, "@;ZLS"))
```

Данное изменение применимо только при использовании функции **Format** для форматирования строки с учетом ее пустоты или значения **Null**. Остальные выражения обрабатываются так же, как и в предыдущих версиях.

Для преобразования приложения из формата Microsoft Access 1.*x* или 2.0 в формат Microsoft Access 97 необходимо изменение программы для обработки значения **Null**.

Пример использования функции DoEvents (Microsoft Access)

В данном примере функция **DoEvents** используется для передачи управления операционной системе каждый раз после выполнения итерации цикла. Функция **DoEvents** всегда возвращает 0.

```
Sub LongLoop()
   Dim intI As Integer

   For intI = 1 To 1500            ' Начало цикла.
      If intI Mod 100 = 0 Then     ' Если цикл выполнен 100 раз.
         DoEvents                  ' Передает управление
                                   ' операционной системе.
         Debug.Print intI
      End If
   Next intI                       ' Увеличивает счетчик цикла.
End Sub
```