Объектно-ориентированное программирование БД. Использование DAO и ADO

Модели объектов Access. Процедурное программирование в DAO. Объект Recordset. Визуальное программирование в Access

Отношения между объектами (объектно-ориентированный подход)

Успех объектно-ориентированного подхода лежит в смещении акцента со структуры данных (в особенности, от вида связей между данными) к процессу, с помощью которого эти данные создаются и уничтожаются. Реальные структуры данных - деталь реализации, лучше всего относящаяся к внутренней работе каждого класса (совокупности объектов). В ООБД в центре разработки оказываются не структуры данных, а процедуры (методы).

Свойства ООБД:

1. **Абстракция:** Каждая реальная "вещь", которая хранится в БД, является членом какого-либо класса. Класс определяется как совокупность свойств (properties), методов (methods), общедоступных (public) и частных (private) структур данных, а также программ, применимых к объектам (экземплярам) данного класса. Классы представляют собой ни что иное, как абстрактные типы данных. Методы - это процедуры, которые вызывается для того, чтобы произвести какие-либо действия с объектом (например, напечатать себя или скопировать себя). Свойства - это значения данных, связанные с каждым объектом класса, характеризующие его тем или иным образом (например, цвет, возраст). Свойства присутствуют не во всех реализациях, по сути дела, они являются краткой записью методов без аргументов (таких как "сообщите свой цвет", "сообщите свой возраст").
2. **Инкапсуляция:** Внутреннее представление данных и деталей реализации общедоступных и частных методов (программ) является частью определения класса и известно только внутри этого класса. Доступ к объектам класса разрешен только через свойства и методы этого класса или его родителей (см. ниже "наследование"), а не путем использования знания подробностей

3. **Наследование (одиночное или множественное):** Классы определены как часть иерархии классов. Определение каждого класса более низкого уровня наследует свойства и методы его родителя, если они только они явно не объявлены ненаследуемыми или изменены новым определением. При одиночном наследовании класс может иметь только один родительский класс (т.е. классовая иерархия имеет древовидную структуру). При множественном наследовании класс может происходить от многочисленных родителей (т.е. иерархия классов имеет структуру ориентированного нециклического графа, не обязательно древовидную). Не все объектно-ориентированные СУБД поддерживают множественное наследование.
4. **Полиморфизм**: Несколько классов могут иметь совпадающие имена методов и свойств, даже если они считаются различными. Это позволяет писать методы доступа, которые будут правильно работать с объектами совершенно различных классов, лишь бы соответствующие методы и свойства были в этих классах определены. Например, метод Print может быть определен во многих классах, но работать по-разному, в зависимости от класса объекта, к которому он применяется.
5. **Сообщения**: Взаимодействие с объектами осуществляется путем посылки сообщений с возможностью получения ответов. Это отличается от традиционного для других моделей вызова процедур. Для того, чтобы применить метод к объекту, надо послать ему сообщение типа "примени к себе данный метод" (например, "напечатай себя"). Парадигма пересылки сообщений не всегда используется в объектно-ориентированных БД, однако типична для "истинно" ОО-реализаций.

Каждый объект, информация о котором хранится в ООБД, считается принадлежащим какому-либо классу, а связи между классами устанавливаются при помощи свойств и методов классов

Модели объектов Access

MS Access представлен двумя уровнями компонентов: ядром БД ***Jet Database Engine*** (ПО **Join* Engine Technology**) и СУБД Access (**Access Application Layer**).

1. ***На уровне ядра*** обслуживаются данные - таблицы и запросы. Ядро транслирует операторы программы в физические операции в БД, читает, записывает и модифицирует БД и индексные файлы, блокирует и объединяет записи и выполняет запросы к БД. Для организации данных используется индексно-последовательный метод (ISAM). Этот уровень поддерживается двумя *библиотеками классов объектов*: ***DAO*** (Data Access Objects) и ***ADO*** (ActiveX Data Objects – ADODB, ActiveX Data Objects Extensions for DDL and Sequrity – ADOX, Microsoft Jet and Replication Objects – JRO, где ADO - это название протокола взаимодействия с БД , а ADODB - название объекта в библиотеке). DAO и ADO являются интерфейсами, используемым для присоединения к Jet Database Engine:
   - **ADODB** обеспечивает приложению доступ к источнику данных с возможностью отбора и изменения данных.
   - **ADOX** позволяет программно изменять структуру объектов источника данных и систему защиты БД.
   - **JRO** служит для создания, модификации и синхронизации реплик БД ***Access***.
2. ***СУБД Access*** обслуживает интерфейс пользователя (формы, отчеты, макросы, меню, панели, окна диалога) и процедуры ***VBA*** (БД можно делить на части и хранить отдельно данные, отдельно – интерфейс).

Jet Database Engine и СУБД Access взаимодействуют друг с другом с помощью языка SQL и библиотек классов объектов DAO и ADO.

Интерфейсы ODBC и OLE BD

1. Интерфейс ODBC (***Open Database Connectivity*** или ***Открытый интерфейс доступа к БД***) был разработан фирмой MS для унифицированного доступа любого приложения к *различным источникам данных* **с помощью языка SQL**. Для подключения к БД с использованием функций ODBC API первоначально следует создать источник данных DSN (*Data Source Name*), в котором задаются его имя и местоположение БД (имя сервера и имя БД на сервере). Интерфейс ODBC используется *библиотекой классов объектов доступа к данным* DAO (*DataAccess Objects* версии 3.6 в Access 2003), к ним относится большинство РБД: MS-SQL server, Oracle, Sybase, DB2 (IBM), SAP.
2. Интерфейс OLE DB (***Object Linking and Embedding, связывание и внедрение объектов*** или ***OLE для БД***) может присоединяться к большинству файлов, в том числе к нереляционным БД, поэтому словосочетание *источник данных* было решено заменить термином *провайдер*. Провайдер OLE DB используется для доступа к данным с помощью *библиотеки классов объектов доступа к данным* ADO (*ActiveX Data Objects версии 2.7 в Access 2003*). Для доступа используется объект ***Connection***, несущий сведения об источнике данных и его расположении и открывающий сеанс обмена данными через провайдер OLE DB.

Jet Database Engine

- Библиотека содержит DLL файлы, присоединяемые к проекту в режиме реального времени.
- Транслирует операторы программы в физические операции в БД.
- Читает, записывает и модифицирует БД и связанные индексные файлы, обслуживает блокировку и объединение записей и выполняет запросы к БД (содержит процессор БД).
- DAO и ADO являются интерфейсами, используемым для присоединения к Jet Database Engine.

ODBC

- Клиент-серверные БД, такие, как MS-SQL server, Oracle, Sybase, IBM - DB2, SAP.

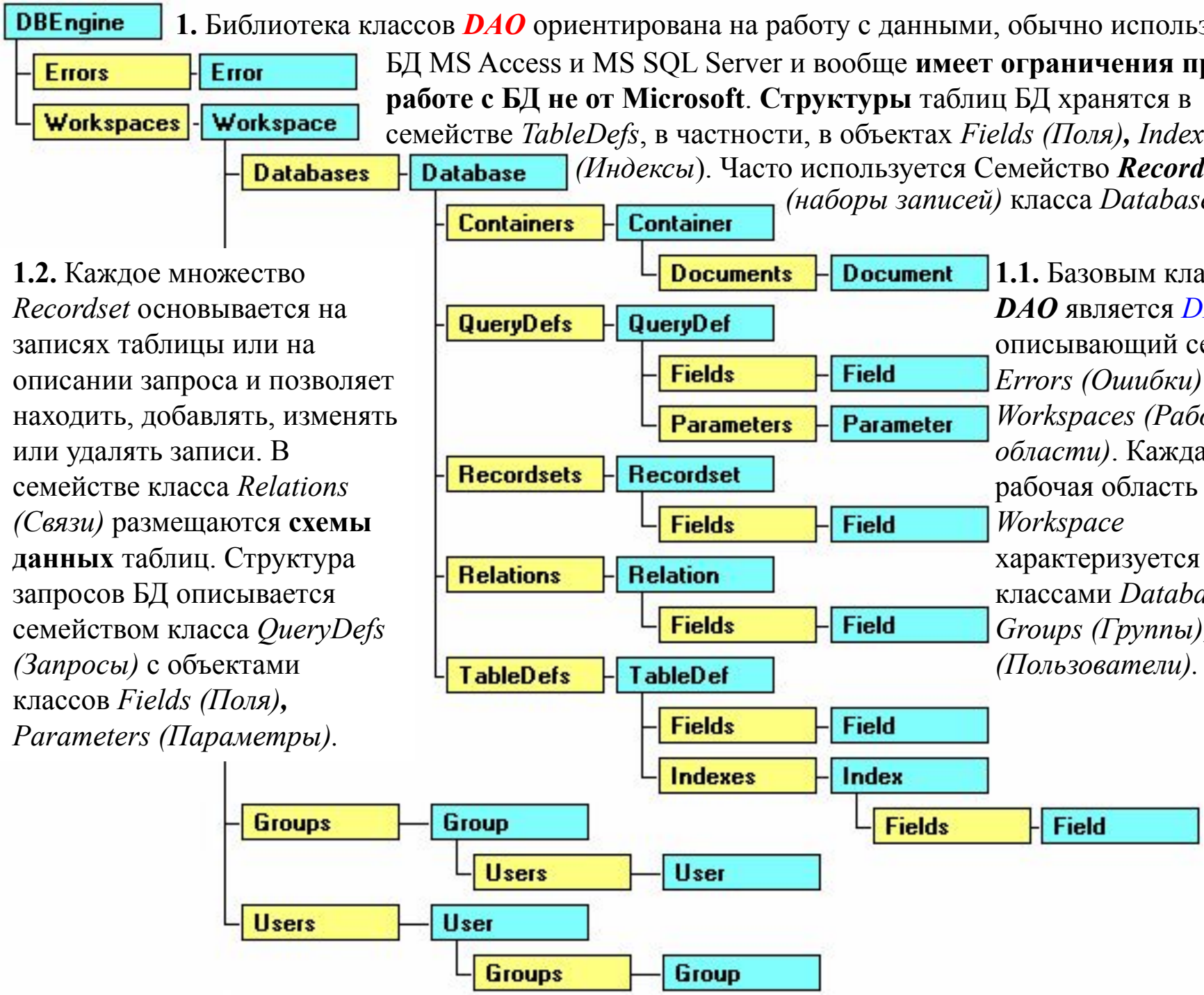
ODBC использует язык SQL для доступа к БД

OLE DB

- Набор COM-интерфейсов для универсального доступа к данным в различных по типам источниках данных (**нереляционных БД**)
- Новый низкоуровневый интерфейс, являющийся частью платформы Universal Data Access
- Позволяет обращаться к курсору (указателю на запись)
- Позволяет обрабатывать пакет записей
- Имеет графические возможности для построения иерархических структур.
- Позволяет обращаться к сетевым БД

ActiveX control

- ActiveX это OLE с расширением для Internet.
- Объекты могут подключаться к форме.
- ActiveX могут добавляться на Web страницы для увеличения их функциональности.
- С их помощью можно проигрывать файлы .avi

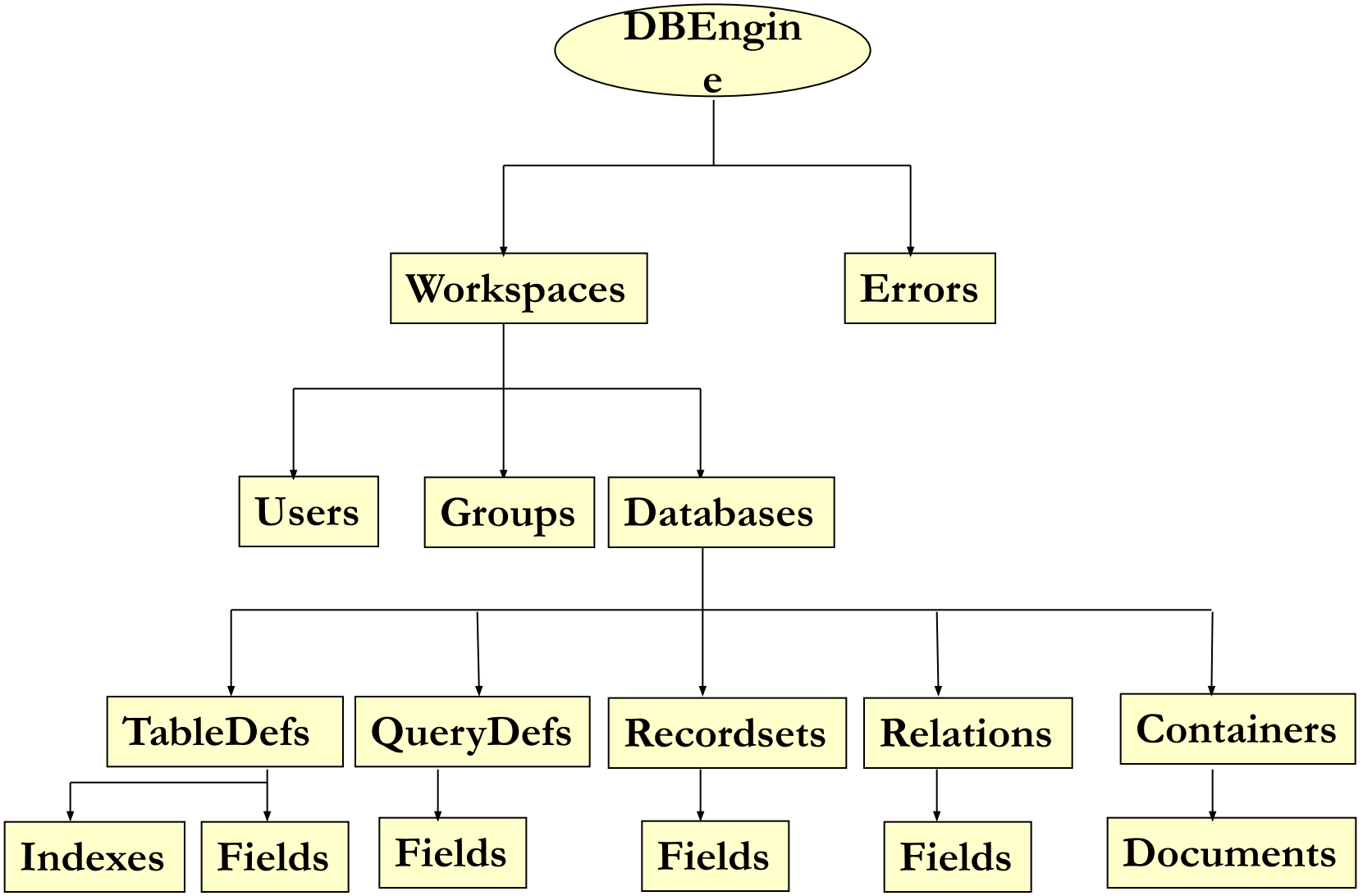


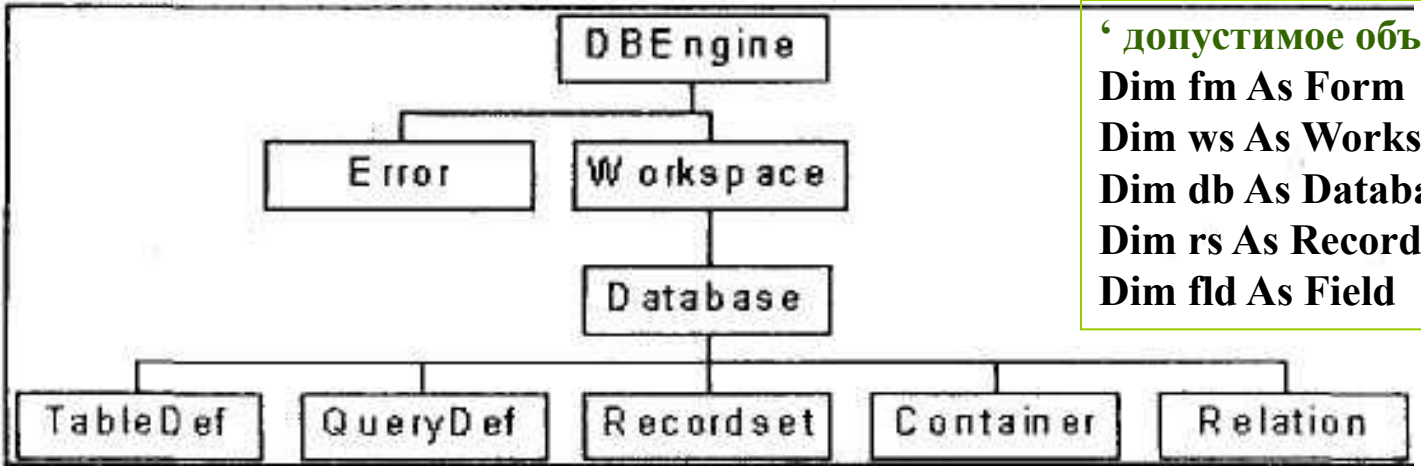
1. Библиотека классов ***DAO*** ориентирована на работу с данными, обычно используется с БД MS Access и MS SQL Server и вообще **имеет ограничения при работе с БД не от Microsoft**. **Структуры** таблиц БД хранятся в семействе *TableDefs*, в частности, в объектах *Fields (Поля)***,** *Indexes (Индексы)*. Часто используется Семейство ***Recordset*** *(наборы записей)* класса *Database*.

1.1. Базовым классом ***DAO*** является *DBEngine*, описывающий семейства *Errors (Ошибки)* и *Workspaces (Рабочие области)*. Каждая рабочая область *Workspace* характеризуется классами *Databases (БД), Groups (Группы), Users (Пользователи)*.

1.2. Каждое множество *Recordset* основывается на записях таблицы или на описании запроса и позволяет находить, добавлять, изменять или удалять записи. В семействе класса *Relations (Связи)* размещаются **схемы данных** таблиц. Структура запросов БД описывается семейством класса *QueryDefs (Запросы)* с объектами классов *Fields (Поля)***,** *Parameters (Параметры)*.

Иерархия объектов Jet Data Base Engine






```
‘ допустимое объявление переменных
Dim fm As Form
Dim ws As Workspace
Dim db As Database
Dim rs As Recordset
Dim fld As Field
```

Объект DBEngine - процессор ядра БД Jet. Объект DBEngine также содержит в себе семейство Workspaces объектов Workspace, которые устанавливают именованные сеансы работы пользователя.

Каждый объект Workspace включает семейство Databases, которое состоит из одного или более объектов Database, являющихся открытыми БД.

Объект TableDef, элемент семейства TabieDefs, представляет сохраненное определение основной или присоединенной таблицы.

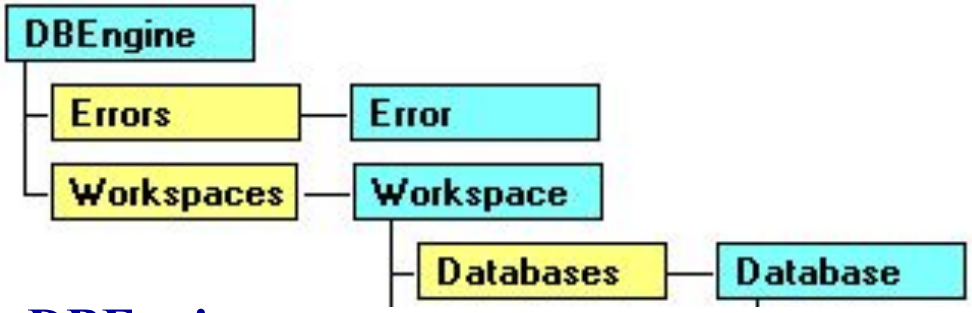
Объект QueryDef, элемент семейства QueryDefs, представляет сохраненное определение запроса в БД Microsoft Jet или временное определение запроса в рабочей области ODBCDirect.

Объект Recordset, элемент семейства Recordsets, представляет набор записей в основной таблице или набор записей, который получается в результате выполнения запроса.

В объектах Container группируются однотипные объекты, а объект Relation представляет связь между полями таблиц или запросами.

Объект Error, являющийся элементом семейства Errors, хранит информацию об ошибках, возникающих при объектном доступе к данным.

Data Access Objects (DAO)



- **DBEngine**

  Находится на вершине иерархии объектов DAO и позволяет обращаться к ядру БД - Microsoft Jet Database Engine.

  Транзакции используются для сохранения (*commit*) или отката (*rollback*) изменений в БД. Методы транзакций в DBEngine:

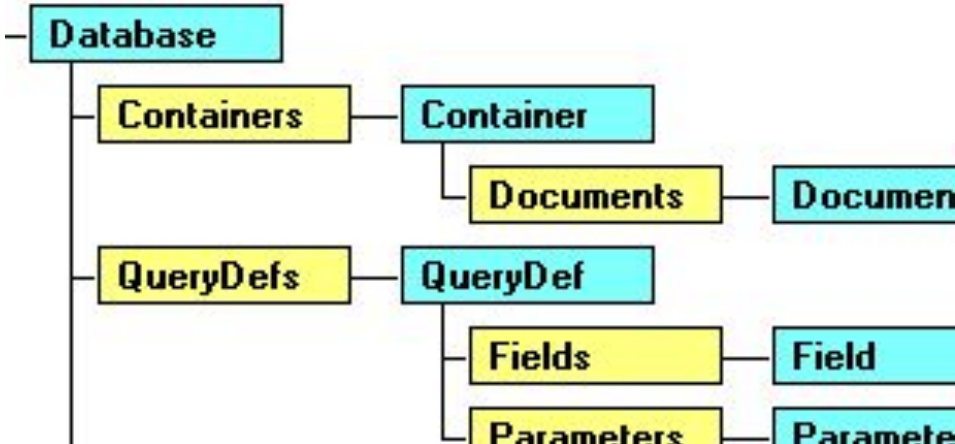
  - ***BeginTrans***
  - ***CommitTrans***
  - ***Rollback***

- **Workspace**

  Поддерживает отдельные транзакции. Позволяет изолировать одну группу транзакций так, что можно выполнить откат.

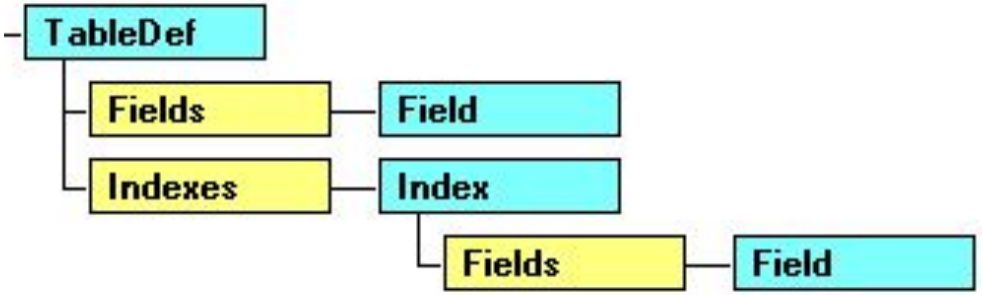
- **Database**

  Содержит объекты TableDefs, QueryDefs, Relations, Recordsets и Containers.



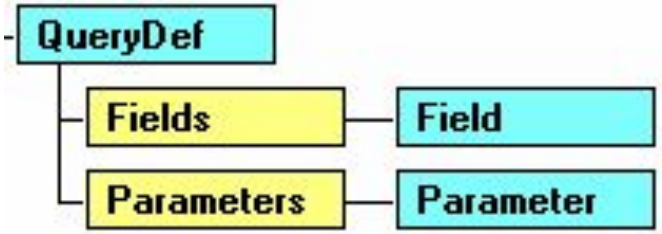
• **TableDef**

Представляет таблицу БД,
т.е. коллекцию полей записей



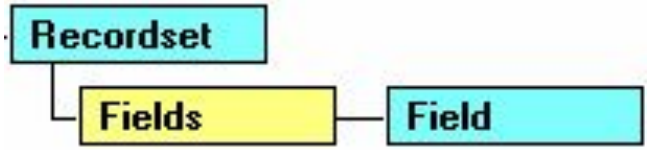
• **QueryDef**

Используется для создания и изменения запросов к БД. Может иметь объекты Field и Parameter (SQL).



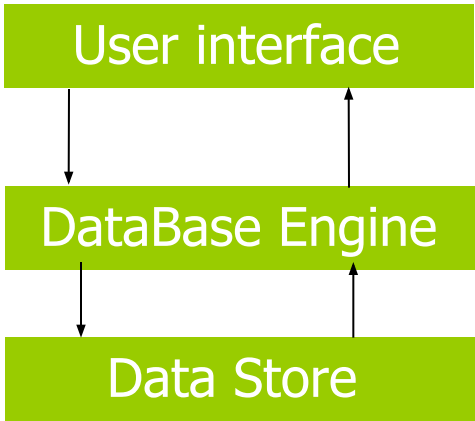
• **Recordset**

Объект табличной структуры, используется для представления данных из объектов TableDef или QueryDef.

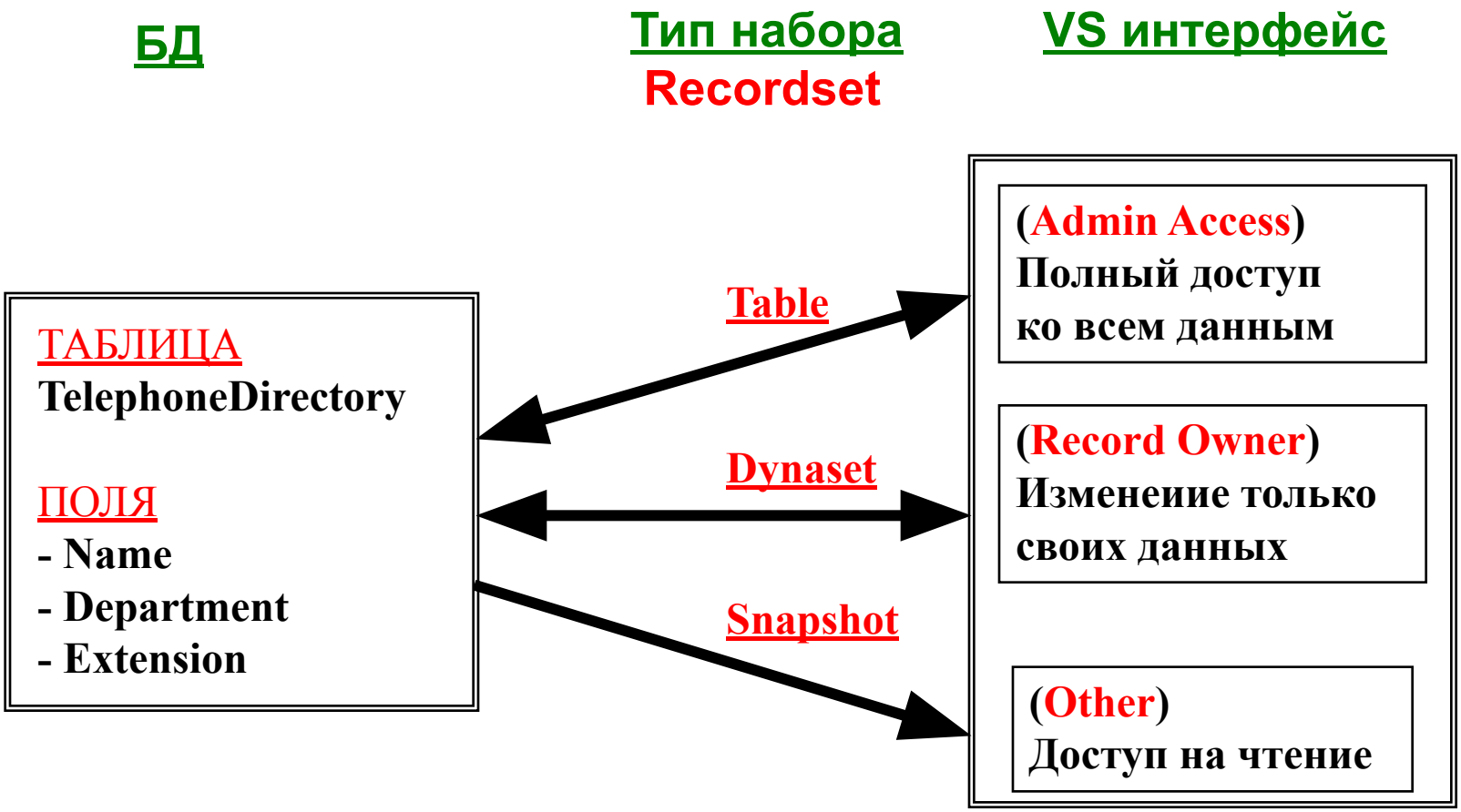


Recordset может быть открыт в режиме:

- Table (чтение и запись) – копия таблицы, медленный метод, связан с нехваткой памяти для всей таблицы
- Dynaset (чтение и запись) – запрос из таблиц или запросов, более быстрый метод
- Snapshot (только чтение) - запрос из таблиц или запросов, самый быстрый метод

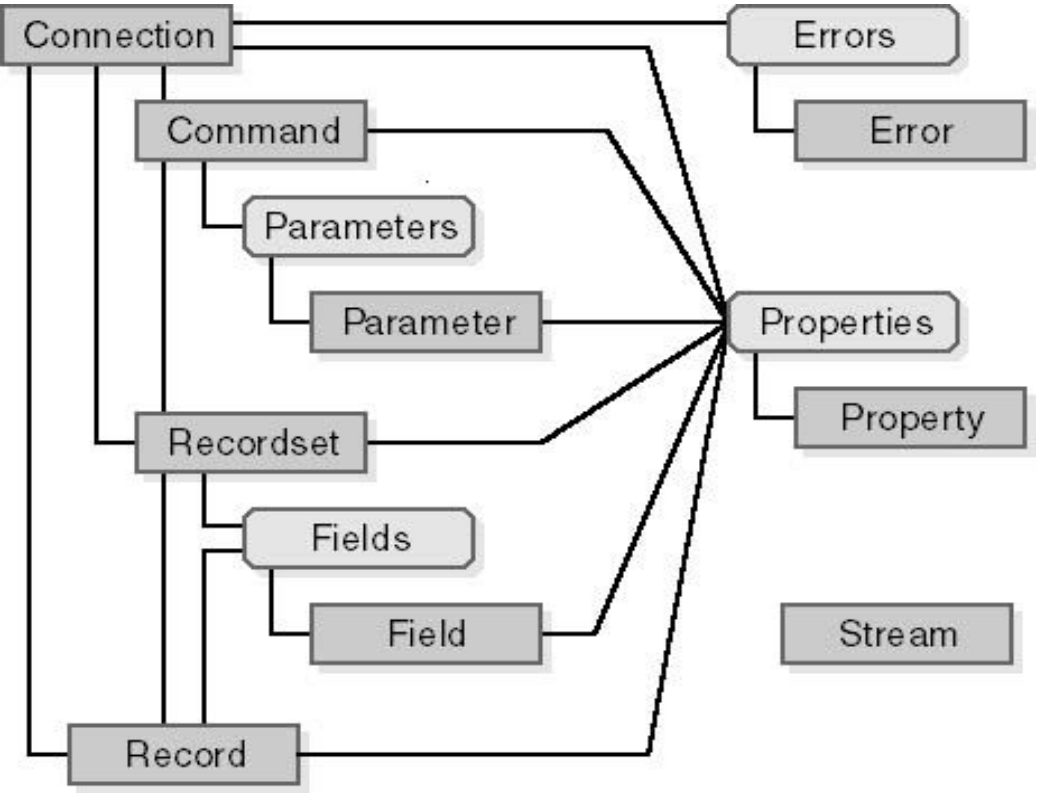


Права на создание набора данных и соответствующие типы набора записей



Семейство	Объект	Описание объекта DAO
-	*DBEngine*	Главный объект ядра БД Microsoft Jet
Connections	*Connection*	Информация о соединении с источником данных ODBC (рабочая область ODBCDirect)
Containers	*Container*	Хранилище информации об объекте (рабочая область MS Jet)
Databases	*Database*	Открытая БД
Documents	*Document*	Информация о сохраненном объекте (рабочая область MS Jet)
Errors	*Error*	Информация об ошибках, ассоциированных с данным объектом
Fields	*Field*	Поле (столбец) таблицы, запроса, индекса, поле связи между таблицами или запросами, поле набора записей
Groups	*Group*	Группа пользователей с правами доступа к данным (рабочая область MS Jet)
Indexes	*Index*	Порядок и уникальность значений в таблице (рабочая область MS Jet)
Parameters	*Parameter*	Параметр для параметризованного запроса
Properties	*Property*	Встроенная или определенная пользователем храктеристика (свойство)
QueryDefs	*QueryDef*	Описание хранимого в БД запроса
Recordsets	*Recordset*	Набор записей в базовой таблице или запросе
Relations	*Relation*	Связь между полями таблиц или запросов (используется в рабочей области MS Jet)
TableDefs	*TableDef*	Описание хранимой в БД таблицы (рабочая область MS Jet)
Users	*User*	Права доступа пользователя к данным (рабочая область MS Jet)
Workspases	*Workspace*	Сеанс работы с источником данных с помощью ядра БД MS Jet

2. Во главе модели ***ADO*** стоит объект *Connection (Соединение),* оно описывает среду обмена данными. Источник данных управляется производным от *Connection* объектом *Command (Команда)*, который командами SQL добавляет, удаляет, обновляет и считывает данные. Его семейство *Parameters (Параметры)* представляет компоненты объекта *Command*. Другой производный от *Connection* объект – ***Recordset*** – накапливает считанные из источника данные. Его семейство *Fields* представляет поля таблиц *Recordset*. Поля характеризуются семействами свойств *Properties*. Встроенные свойства являются частью объекта ***ADO*** и всегда доступны, а динамические свойства существуют только в момент работы источника данных.



ADO содержит семь объектов:
Connection, Command, Parameter, Recordset, Field, Property, Error (Stream – новый*)*
и **четыре семейства (коллекции) объектов:**
Fields, Properties, Parameters, Errors

```
Set Con = New ADODB.Connection
Con.Open “Provider='sqloledb';Data Source='SqlServer';Initial Catalog='Pubs';” & _
Integrated Security='SSPI';”
Dim rst As New ADODB.Recordset
strSQL = “…"
rst.Open strSQL, Con, adOpenKeyset, adLockBatchOptimistic, adCmdTable

rst.MoveFirst
Do Until rst.EOF
    …
    rst.MoveNext
Loop

If MsgBox("Save all changes?", vbYesNo) = vbYes Then
    rst.UpdateBatch
Else
    rst.CancelBatch
End If
```

Объект	Тип	Описание объекта модели ADO
Connection	Объект	Сеанс обмена данными
Command	Объект	Инструкция SQL
Parameters	Семейство	Параметр инструкции SQL
Recordset	Объект	Набор записей, осуществляет навигацию по записям и манипуляцию с данными в наборе
Fields	Семейство	Поле (столбец) в наборе записей Recordset
Errors	Семейство	Ответ на одну ошибку, произошедшую во время сеанса связи
Property	Объект	Характеристика (свойство) любого объекта ADO
Properties	Семейство	Свойства объектов Connection, Command, Recordset или Field

Объект	Тип	Описание объекта модели ADOX
Catalog	Объект	Главный объект модели. Доступ к таблицам, представлениям, процедурам, группам, пользователям
Tables	Семейство	Таблицы в источнике данных. Каждый объект Table семейства ссылается на одну таблицу
Indexes	Семейство	Индексы таблицы. Каждый объект Index семейства ссылается на один из индексов
Keys	Семейство	Все ключи таблицы. Каждый объект Key семейства ссылается на один из ключей
Columns	Семейство	Оъекты Column, которые ссылаются на столбцы водном из объектов Table, Index, Key
Groups	Семейство	Каждый из объектов Group ссылается на бюджет группы в каталоге или пользователе
Users	Семейство	Бюджеты пользователей, имеющих права доступа к защищенной БД
Procedures	Семейство	Хранимые процедуры в БД. Каждый объект procedure семейства ссылается на одну из хранимых процедур
Views	Семейство	Содержит все представления (View) в БД

Метод **Clone** объекта Recordset (в DAO и ADO) создает копии наборов записей. При копировании организуется указатель текущей записи, а закладки копии совпадают с закладками оригинала. Копия не наследует свойств *Index, Filter, Sort*.

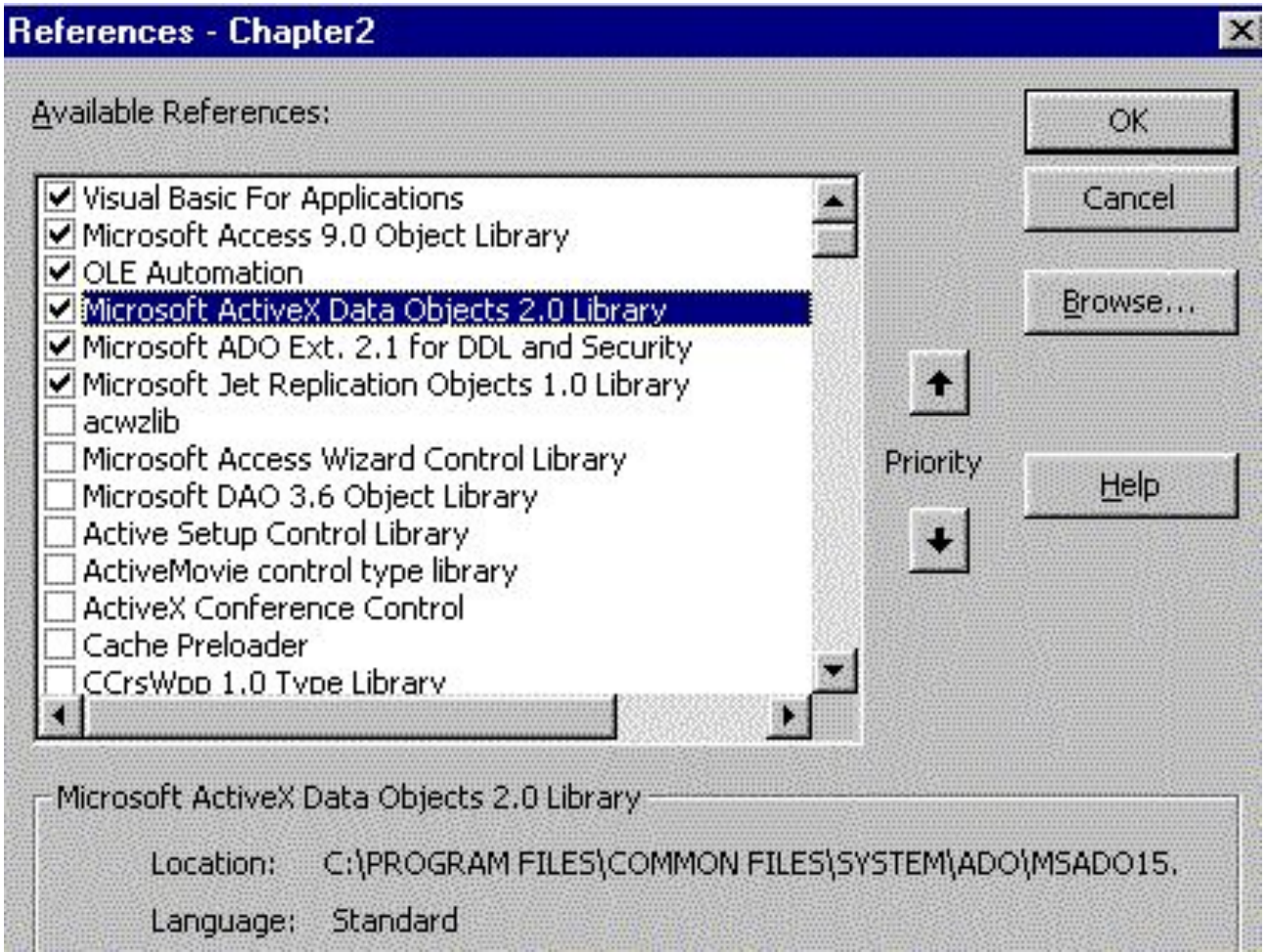
```
Dim db As Database
Dim Products(1 To 3) As Recordset
Set db= OpenDatabase("Northwind.mdb")
Set Products(1)=db.OpenRecordset & _
("SELECT ProdName FROM Products ORDER BY ProdName", dbOpenSnapshot)
Set Products(2) = Products(1).Clone
```

Метод Relations (DAO) объекта Database описывается связь полей таблиц и запросов. При этом указываются поля в первичной и внешней таблицах. В первой таблице эти поля образуют PK, а во второй таблице – FK. Каждое поле первичного ключа следует добавить в семейство **Fields** связи с указанием внешнего ключа в свойстве *ForeignName.* Access устанавливает параметр ссылочной целостности, но не устанавливает каскадное обновление и удаление записей.

```
Set объектСвязи = БД.CreateRelation ([«Имя», Таблица, внешняяТаблица, Атрибуты])
объектСвязи.Table = «Таблица»
объектСвязи.ForeignTable = «внешняяТаблица»
Set объектПоле = объектСвязи.CreateField(«Поле»)
объектПоле.ForeignName = «Поле»
объектСвязи.Fields Append объектПоле
БД.Relations Append объектСвязи
```

Подключение библиотек объектов

Для использования объектов доступа необходимо задать ссылку на библиотеки DAO и/или ADO в редакторе VBA, команда СЕРВИС - ССЫЛКИ (TOOLS - REFERENCES), в окне ССЫЛКИ (REFERENCES) установить флажки. Библиотека, что указана в списке раньше, будет использоваться по умолчанию, или надо явно указывать необходимую библиотеку (DAO или ADO) в коде VBA.



Определение типа набора данных, возвращаемого свойством Recordset формы:

```
Sub CheckRSType()
        Dim rs as Object
        Set rs=Forms(0).Recordset
        If TypeOf  rs Is DAO.Recordset Then
              MsgBox "DAO Recordset"
        ElseIf TypeOf rs Is ADODB.Recordset Then
              MsgBox "ADO Recordset"
        End If
End Sub
```

Вы будете работать в среде DAO в Access 2003 в нескольких ситуациях:

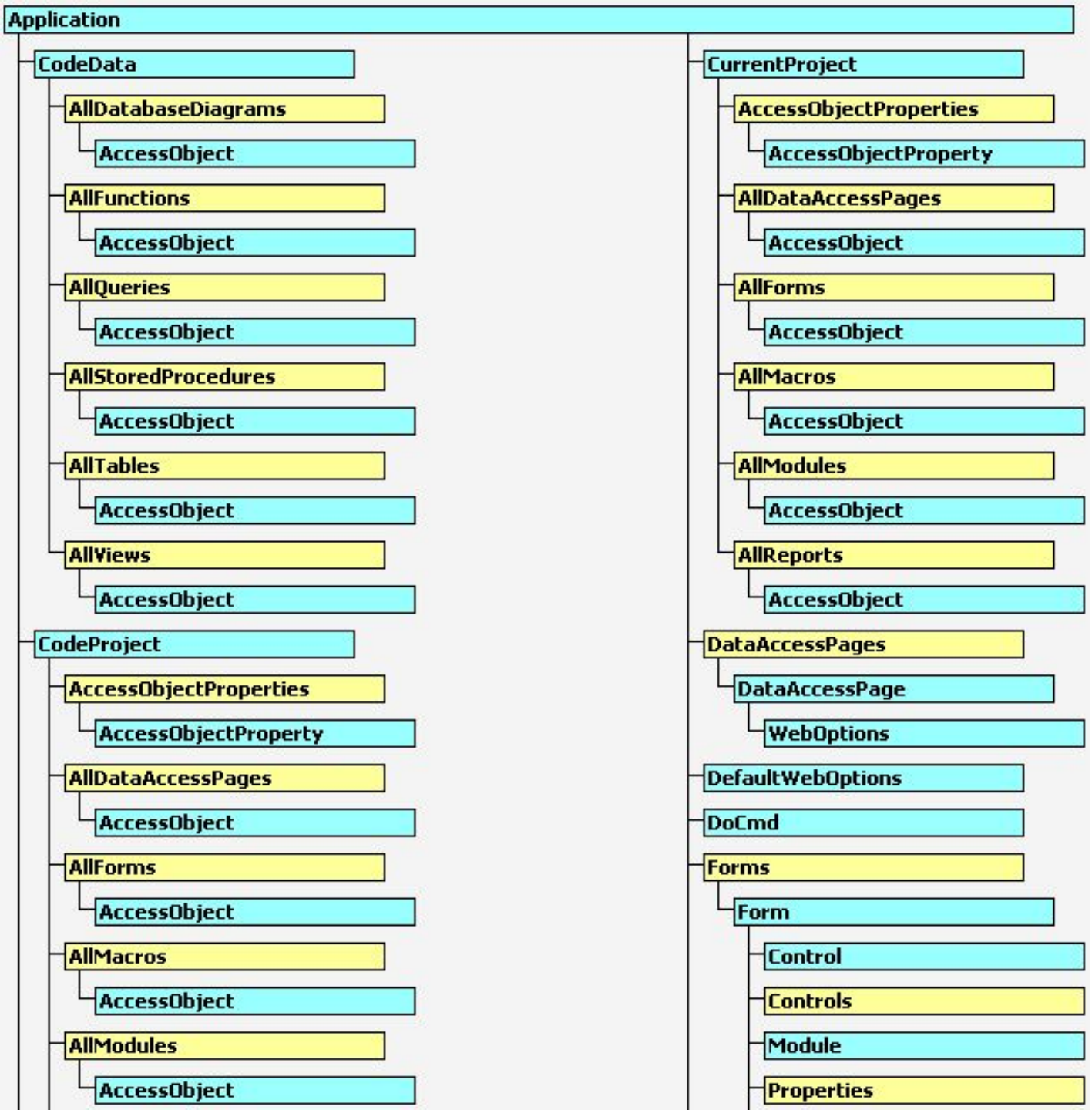
. Если извлекаете recordset формы в БД Access (.mdb), вы получите DAO recordset. Формы в проектах (.adp) используют ADO Recordset.

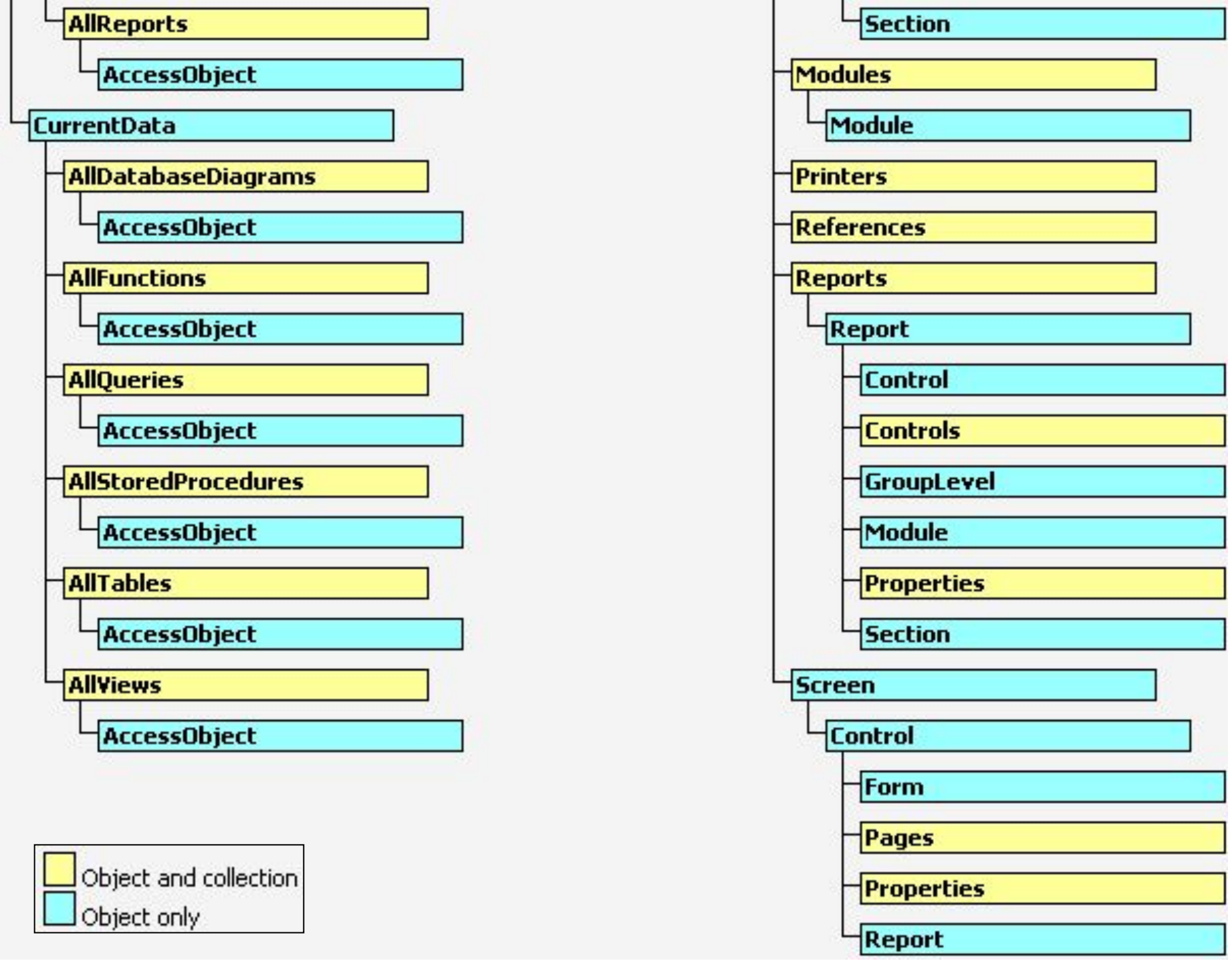
. Если преобразуете БД из предшествующей версии Access. Если создаете новую БД в Access 2003, используете библиотеку ADO по умолчанию.

Когда создаете новую БД (.mdb) или проект (.adp), Access допускает работу с ADO и устанавливает ссылку на MS ActiveX Data Objects Library.
Библиотеки DAO и ADO имеют объекты с одинаковыми именами. Если выбираются обе библиотеки в пределах приложения, то ссылка будет выполнена на объект библиотеки, указанной первой в списке REFERENCES. Можно изменить порядок ссылок или в строке кода прямо сослаться на библиотеку:

```
Dim rst1 As ADODB.Recordset
Dim rst2 As DAO.Recordset
```

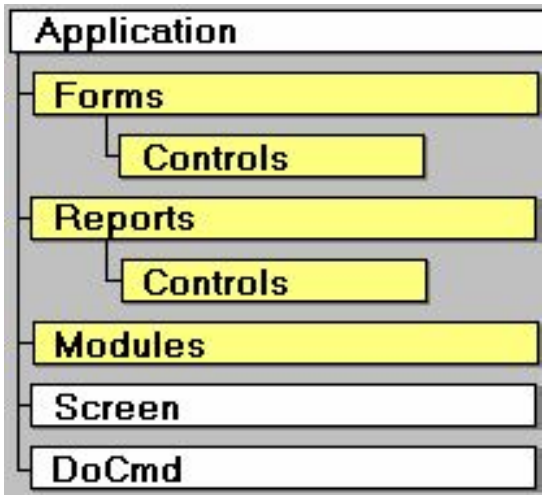
Модель объектов СУБД MS Access

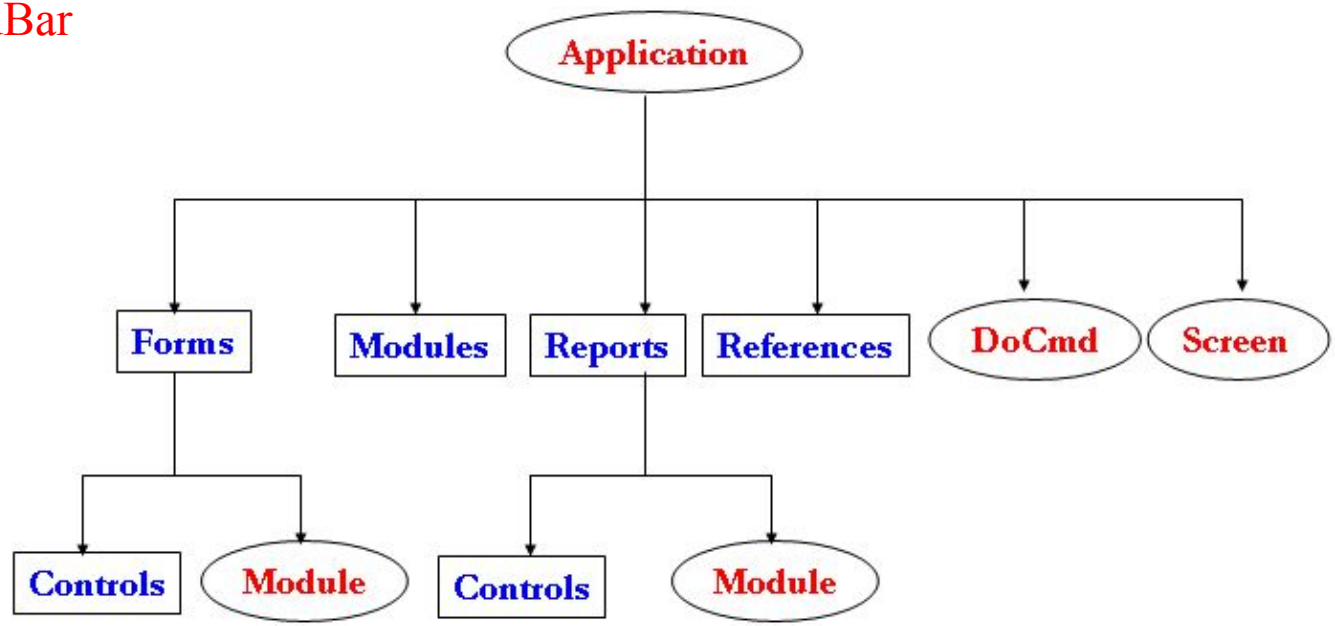



AllReports
AccessObject
CurrentData
AllDatabaseDiagrams
AccessObject
AllFunctions
AccessObject
AllQueries
AccessObject
AllStoredProcedures
AccessObject
AllTables
AccessObject
AllViews
AccessObject
Section
Modules
Module
Printers
References
Reports
Report
Control
Controls
GroupLevel
Module
Properties
Section
Screen
Control
Form
Pages
Properties
Report
Object and collection
Object only

Визуальное программирование в Access

Для работы с формами и отчетами используется библиотека классов объектов СУБД Access с базовым классом Application (Приложение), описывающим семейства (коллекции) Forms, Reports, Modules, References, DataAccessPages, Controls, и объектами Screen, DoCmd, Module, Assistant, CommandBar





Обращение к объектам в иерархии выполняется перечислением всех ее элементов для отслеживания вложенности объектов. Для разделения коллекции объектов и следующего за ним экземпляра коллекции используется восклицательный знак (!). Для разделения объекта и следующего за ним элемента его коллекции используется знак точка (.)

Application. Forms! NameOfForm

или **Forms! NameOfForm**

Имя формы, содержащее знак “пробел”, надо заключить в квадратные скобки

Application. Forms! [Name Of Form]

Итак, схема обращения к элементам библиотеки классов Access:
имяКласса! имяОбъекта[.имяЭлементаУправления] .Элемент

Восклицательный знак используется для перечисления семейств и объектов в иерархии семейств вплоть до появления свойств и методов объекта.
Точка отделяет объект от его свойства или метода, а также разделяет объекты подчиненной формы (отчета):
[Forms!Форма!][Объект].[свойствоИлиМетод]
[Reports!Отчет!][Объект].[свойствоИлиМетод]
[Forms!ГлавнаяФорма!]ПодчиненнаяФорма.[Form!]Объект
[Forms!ГлавнаяФорма!]ПодчиненнаяФорма.eeИсточникДанных

Обращение к объектам форм и отчетов выполняется по схемам:
[Forms! Форма!] Объект
[Reports! Отчет!] Объект
Forms («имяОбъекта»)
Reports («имяОбъекта»)
Forms (индексОбъекта)
Reports (индексОбъекта)
Forms (ссылочнаяПеременная)
Reports (ссылочнаяПеременная)

Объект	Тип	Описание объекта Access
Application	**Объект**	Активное приложение Access
Forms	Семейство	Все открытые формы Access
Pages	Семейство	Все вкладки формы (Tab Control). Является свойством объекта Control
Section	**Объект**	Раздел формы или отчета: заголовок, примечание, колонтитулы, раздел данных
Reports	Семейство	Все открытые отчеты Access
GroupLevel	**Объект**	Массив, содержащий уровни группировки в отчете - GroupLevel (n)
Modules	Семейства	Все открытые стандартные модули и модули объектов Access
Screen	**Объект**	Ссылка на форму, отчет или элемент управления, который в данный момент имеет фокус
DoCmd	**Объект**	Запуск макроса или инструкции Access с помощью VBA
Controls	Семейство	Все элементы управления формы, отчета или секции (Checkbox, TextBox, ComboBox, CommandButton, CustomControl, BoundObj ect Frame, Image, ListBox, ObjectFrame, OptionButton, OptionGroup, Page, Section, SubForm, TogleButton)
DataAccessPages	Семейство	Все открытые страницы доступа к данным Access
References	Семейство	Все ссылки на внешние библиотеки и проекты
DBEngine	**Объект DAO**	Объект самого верхнего уровня модели доступа к данным DAO.
CurrentProject	**Объект**	Программный проект . Семейства объектов AccessObjects: AllForms, AllReport AllMacros, AllModules, AllDataAccessPages, независимо от того, открыты они или закрыты в данный момент
CurrentData	**Объект**	Объекты, сохраненные источником данных (ядром Jet и SQL-сервером) в текущей БД: AllTables, AllQueries, AllViews, AllStoredProcedures, All Functions, AllDatabaseDiagrams, независимо от того открыты они или закрыты в данный момент

Основными методами класса **Application** являются:

Метод	Назначение
CloseCurrentDatabase	Закрывает текущую БД из другой базы
NewCurrentDatabase	Создает новый объект Database из другого приложения
OpenCurrentDatabase	Открывает в качестве текущей БД объект Database
Quit	Закрывает Access
Run	Запускает процедуру Access из другого приложения
RunCommand	Запускает команду меню или панели инструментов

Семейство Forms *(Формы)* содержит формы и их объекты, принадлежащие классу Controls *(Элементы управления)*.

Семейство Reports *(Отчеты)* включает все отчеты приложения и объекты отчетов того же класса *Controls*.

Семейство Modules *(Модули)* объединяет все стандартные модули и модули авторских классов, а также модули, связанные с формами и отчетами.

Класс объектов **Screen** используется для работы с окнами.

Класс объектов **DoCmd** позволяет обращаться из модулей к стандартным средствам Access, т.е. дублировать операции интерфейса Access. Методы объекта DoCmd не возвращают значений. Аргументами являются константы Access: acTable, acQuery, acForm, acReport, True, False. Имена объектов помещаются в двойные кавычки. Для выполнения запросов ***SQL*** вызывается метод RunSQL объекта DoCmd.

DoCmd.OpenForm Объект[,режимВывода][,Запрос][,условиеОтбора][,режимДанных][, режимОкна][, аргументыОткрытия]

DoCmd.OpenReport Объект[,режимВывода][,Запрос][,условиеОтбора][,режимДанных] [, режимОкна][,аргументыОткрытия]

Обращение к активным объектам класса Screen заменяют их свойствами ActiveDataSheet, ActiveForm, ActiveControl, PreviousControl, Application, Parent.
Для ссылки на активную форму или отчет из модулей их классов можно так:
Me.Элемент
Me!Объект

Закрытие объектов выполняется методом Close класса Access и объекта DoCmd, а выход из Access – методом Quit:
DoCmd.Close [типОбъекта, имяОбъекта], [параметрСохранения]
Application.Quit [параметрСохранения]
DoCmd.Quit [параметрСохранения]
где *типОбъекта* и *параметрСохранения* выражаются встроенными константами.

```
Sub ОткрытиеФормыAccess()
    DoCmd.OpenForm "Форма Таблицы1"
    MsgBox Forms![Форма Таблицы1].Caption  ‘ титул окна формы
End Sub

Sub ОткрытиеФормыAccess()
    On Error GoTo Ошибка                ‘ обработка ошибки
    Имя = InputBox ("Какую открыть форму?")
    DoCmd.OpenForm Имя
    Exit Sub
    Ошибка:
        MsgBox "Ошибка " & Err, vbExclamation
        Exit Sub
End Sub
```

Параматры метода OpenForm:

OpenForm(*FormName*, *View*, *FilterName*, *WhereCondition*, *DataMode*, *WindowMode*, *OpenArgs*)

```
Sub СвойстваФормыAccess ()
DoCmd.OpenForm FormName:="Мы", view:=acDesign
With Forms!Мы
        .Caption = "Сведения о клиентах"
                .Modal = True
                .DefaultView = 1
                .AllowEdits = False
                .AutoResize = False
                .AutoCenter = True
                 .Width = 300
                .Picture = "d:\f1.bmp"
                .Tag = "0"
                .OnActivate = "=FunctionName ()"
                .OnCurrent = "[Процедура обработки событий]"
                .OnClose = "Макрос"
End With
DoCmd.OpenForm "Мы"
End Sub

Sub ОткрытиеОтчетаAccess ()
        DoCmd.OpenForm "Выставка"
        ответ = MsgBox ("Вывести отчет ?", vbYesNoCancel)
        If ответ = vbYes Then
                DoCmd.OpenReport "Выставка", acPreview
        MsgBox Screen.ActiveReport.Name
End Sub
```

acDesign
acFormDS
acFormPivotChart
acFormPivotTable
acNormal *default*
acPreview

Немодальный режим открытия окна - допускается переход в другие окна. Модальный режим - форма или отчет сохраняет фокус до своего закрытия.

Свойства объектов Form и Report

Свойство	Тип	Описание
ActiveControl	ro, o	Активный элемент управления открытой формы
Count	ro, i	Число открытых элементов семейства
CurrentRecord	ro, i	Номер текущей записи формы
Cycle	i	AllRecords, CurrentRecord или CurrentPage в форме
Dirty	ro, b	Изменена ли текущая запись после сохранения
Form	ro, o	Ссылка на форму
HasData	ro, i	В отчете есть данные (-1) или нет данных (0)
Me	ro, o	Сама форма или отчет
MenuBar	s	Имя строки меню или стандартное меню («»)
Modal	b	Модальная ли форма
NewRecord	ro, b	Является ли текущая запись формы новой
OrderBy	s	Список полей, определящий порядок сортировки
Painting	b	Необходимо ли перерисовать форму
Picture	bmp	Имя файла фона
PopUp	b	Всплывающая ли форма
RecordSetClone	ro, o	Доступ к свойствам записей RecordSource
RecordSource	s	Источник данных: таблица или запрос
Report	ro, o	Ссылка на отчет
Section	ro, o	Доступ к области или ее элементу управления
ShortcutMenuBar	s	Имя контекстного меню или стандартное («»)
StatusBarText	s	Текст строки состояния формы
TabIndex	i	Номер элемента управления формы
ToolBar	s	Имя панели инструментов или стандартная («»)
Visible	b	Отображается ли на экране

o – объект,
i – целый,
b – логический,
s – строковый,
ro – только чтение

```
Sub ЗакрытьФормуОтчет ()
    DoCmd.OpenForm “Сотрудники“,,, “Должность = “””дилер””” 
    MsgBox Forms (“Сотрудники“).Caption
    DoCmd.Close acForm, "Сотрудники", acPrompt
    DoCmd.OpenReport “Сотрудники“, acViewPreview
    MsgBox Reports!Сотрудники.Page
    DoCmd.Close Save:=acSaveYes
End Sub

Function КонецРаботыAccess ()
    ответ = MsgBox ("Закончить работу?", vbYesNoCancel + vbQuestion)
    Select Case ответ
        Case vbYes
            DoCmd.Quit acPrompt
            КонецРаботыAccess = True
        Case vbNo
            DoCmd.Close acForm, "Форма Таблицы1", acSaveYes
            КонецРаботыAccess = False
        Case vbCancel
            КонецРаботыAccess = False
    End Select
End Function
```

OpenArgs – установлен фильтр на значение поля ДОЛЖНОСТЬ

Методы объекта Form

Метод	Действие
GoToPage	Передает фокус первому элементу управления активной формы
Recalc	Обновляет вычисляемые элементы, но не функции SQL
Refresh	Обновляет записи без учета добавлений и удалений
Requery	Обновляет источник данных формы (учет добавлений и удалений). Эквивалентно повторному открытию формы
SetFocus	Передает фокус элементу, ранее последним им владевшему
Undo	Отменяет изменения в форме
Repaint	Обновляет экран и завершает вычисления

Обновление объектов, активных таблиц и запросов выполняется методом Requery:

имяОбъекта.Requery

DoCmd.Requery

```
Sub ПолеФормыAccess ()
    Dim поле As Control
    DoCmd.OpenForm "Клиенты"
    Set поле = Forms!Клиенты![Обращаться к]
    поле = "Сергей Петров"
    Forms!Клиенты.Refresh ‘ данные обновляются только на форме
End Sub
```

Событие *До обновления* (BeforeUpdate) возникает в момент перемещения с записи на запись. Если пользователь изменяет запись, свойство формы *Dirty* становится истинным, и если после этого аргументу *Cancel* присвоить значение *True,* событие отменяется. Метод *Undo* восстанавливает исходные данные.
Событие *После обновления* (AfterUpdate) используется для выполнения определенных действий в зависимости от введенных в поле значений.

При передаче фокуса от одного элемента управления к другому возникают события *Вход* (*Enter*)**,** соответствующее приему фокуса, и *Выход* (*Exit*), соответствующее его потере. Для выявления идентичных объектов можно использовать оператор **Is**. Он определяет логический результат как Null или Not Null.

Для ссылок **из формы или отчета** на объект *Recordset* используется свойство RecordSetClone. Оно определяет копию записей базовой таблицы или запроса, указанных в свойстве *Источник записей* формы. В частности, если форма основана на запросе, то обращение к свойству *RecordsetClone* эквивалентно созданию копии объекта *Recordset* с помощью того же запроса.

```
Sub СвободныеЭлементыAccess()
    Dim надпись As Object, поле As Object
    DoCmd.OpenForm "Форма1"
    Set надпись = Forms!Форма1.Надпись1
    Set поле = Forms!Форма1.Поле0
    надпись.Caption = поле.Text
End Sub
```

Надпись на метке, кнопке, заголовке формы или отчета

```
Sub ОбновлениеЧерезФормуAccess ()
    DoCmd.OpenForm "Сотрудники", , , "Фамилия = 'Иванов'"
    Set где = Forms!Сотрудники
    If IsNull (где.оклад) Then где.оклад = 0
    где.оклад = InputBox ("Можно заменить " & Str (где.оклад), , где.оклад)
    DoCmd.Requery
End Sub

Sub СтильПоляФормы ()
    DoCmd.OpenForm "Клиенты", acDesign, , , , acHidden
    Set поле = Forms!Клиенты!Имя
    поле.FontSize = 14
    поле.FontName = "Times New Roman Cyr"
    поле.Enabled = False
    DoCmd.Save acForm, "Клиенты"
    DoCmd.OpenForm "Клиенты"
End Sub

Sub НадписьAccess ()
    Set форма = Forms![Сотрудники]
    With форма
        .[ОкладНадпись].Caption = "Должность"
        .Controls![ОкладНадпись].FontName = "Arial Cyr"
        .Оклад.SetFocus
        .Оклад.Requery
        .Дата.Enabled = False
    End With
End Sub
```

```
Sub Form_Current ()
p = Me![Поле1] Like "К*"  ‘логическое значение
      If  p Then
            MsgBox "Кузнецов!"
      Else
            MsgBox "Не он"
End Sub

Sub Кнопка_ Click ()
      On Error GoTo 1
      Err.Clear
      With Me
            !Премия.SetFocus
            память = !Премия.Text
            !Оклад.SetFocus
            !Оклад.Text = Val (!Оклад.Text) + Val (память)
            .Refresh
      End With
      Exit Sub
1:
      MsgBox "Ошибка номер " & Err
End Sub
```

```
Sub КнопкаОшибки_Click ()
        On Error GoTo 1
        Err.Clear
        n = InputBox ("Введите число")
        Me!Оклад.SetFocus
        Me!Оклад.Text = n
        Me.Refresh
        Exit Sub
1:
        m = "Ошибка № " & Str (Err.Number) & " возникла в " & Err.Source & Chr (13)
& _             Err.Description
MsgBox m, , "Ошибка"
End Sub

Sub ФормаAccess_BeforeUpdate (Cancel As Integer)
If Me.Dirty Then ‘ данные на форме изменились
        If MsgBox ("Сохранить изменения?", vbYesNo) = vbNo Then
                Me.Undo
        End If
End Sub

Sub ФормаAccess_AfterUpdate ()
        MsgBox “в “ & Me.Name & “кое-что изменилось“
End Sub
```

```
Sub НавигацияAccess()
DoCmd.OpenForm "Сотрудники"
Set форма = Forms!Сотрудники
DoCmd.GoToRecord  , , acLast
форма.Надпись.Caption = Str(форма.CurrentRecord)
DoCmd.GoToRecord  , , acNewRec              ‘ новая запись
форма.Фамилия = InputBox ("Следующий?")
форма.Refresh
End Sub
‘ DoCmd.GoToRecord acActiveDataObject(default), ObjectName, acNext(default), Offset

Function ПодчиненнаяФормаAccess()
DoCmd.OpenForm "Главная"
DoCmd.GoToControl "Подчиненная форма"
DoCmd.GoToRecord  , , acNewRec
With Forms!Главная
      ![Подчиненная форма]!ПодчТовар = !КлиентыТовар
      ![Подчиненная форма]!ПодчНазвание = !КлиентыНазвание
      ![Подчиненная форма]!ПодчЦена = !КлиентыЦена
End With
Forms!Главная.Refresh
End Function
```

```
Sub Дни_Click ()
On Error GoTo label0
Dim день(1) As Date, ошибки As Integer, поле As Object
Set поле = Me!дата
DoCmd.GoToRecord  , , acFirst
Do While 1
        день (0) = поле
        DoCmd.GoToRecord  , , acNext
        день (1) = поле
        If DateDiff ("d", день (0), день (1)) <> 1  Then
                MsgBox день (0) & " или " & день (1) & " неверно" ошибки = ошибки + 1
        End If
Loop
label0:
MsgBox "Всего ошибок: " & Str (ошибки)
End Sub

Sub ОткрытыеОтчеты ()
список = "Открыто отчетов: " & Reports.Count
For Each отчеты In Reports
        список = список & Chr (13) & отчеты.Name
        For Each объекты In отчеты.Controls
                список = список & Chr(13) & "Объект " & объекты.Name
        Next объекты
Next отчеты
MsgBox список
End Sub
```

```
Sub ЗапросSQL ()
DoCmd.OpenForm "Сотрудники"
DoCmd.RunSQL “UPDATE Сотрудники SET Родился = #01/12/02# WHERE Номер = 28“
DoCmd.RunSQL “DELETE FROM Сотрудники WHERE Родился = #01/12/02#“
DoCmd.RunSQL “INSERT INTO Сотрудники (ФИО, Родился, Должность) VALUES (”Кто-то‘, #01/12/12#, ”Дилер‘)“
DoCmd.Requery
End Sub

Sub ПеренумероватьЗаписи ()
DoCmd.RunSQL "ALTER TABLE Сотрудники DROP COLUMN z"
DoCmd.RunSQL "ALTER TABLE Сотрудники ADD COLUMN z COUNTER"
DoCmd.OpenForm "Сотрудники"
End Sub

Sub ЗапросAccess ()
DoCmd.SetWarnings False
DoCmd.OpenForm "Сотрудники", acDesign, , , , acHidden
DoCmd.OpenQuery "Запрос1"
DoCmd.Close acForm, "Сотрудники"
DoCmd.SetWarnings True
End Sub
```

```
Sub АнализТаблицыЧерезФорму ()
DoCmd.OpenForm "Выставка"
MsgBox Forms (“Выставка“).RecordsetClone.RecordCount & " записей "
End Sub

Sub AccessForm_Open ()
If Me.RecordsetClone.RecordCount = 0 Then
        MsgBox "Записей нет", vbInformation
Else
        MsgBox Me.RecordsetClone.RecordCount & " записей"
End If
End Sub

Sub ФормаADO ()
Dim cnn As New ADODB.Connection, rst As New ADODB.Recordset
cnn.Open "DBQ=D:\Сотрудники.xls; Driver={Microsoft Excel Driver (*.xls)}"
rst.CursorType = adOpenStatic
rst.Open Лист1, cnn
DoCmd.OpenForm “Form1“
Set Forms (“Form1“).Recordset = rst
Forms (“Form1“).Controls ("Надпись").Caption = rst.RecordCount
Forms (“Form1“).Controls ("Список").RowSource = "select * from rst.Source"
End Sub
```

Курсор

Курсор – текущая запись. Типы курсора:

- **adOpenStatic**. Изменения, внесенные другими пользователями – невидимы
- **adOpenForwardOnly**. Подобен первому типу, однако можно передвигаться только вперед по записям (быстрый метод)
- **adOpenDynamic**. Чужие изменения записей отображаются, перемещение по записям в любом направлении.
- **adOpenKeyset**. Как третий тип, но не отображаются добавленные записи.

Программное блокирование записей

Для исключения возможности внесения изменения в запись, редактируемую другим пользователем, каждый пользователь должен заблокировать запись перед ее редактированием. Типы блокировки:

- **adLockReadOnly** - чтение записей без их блокировки.
- **adLockPessimistic** – запись блокируется при получении доступа к ней и освобождается при переходе к другой записи.
- **adLockOptimistic** – запись блокируется в момент ее изменения (update)

  recordset**.Update** Fields**,** Values

  record.Fields**.Update**
- **adLockBatchOptimistic** - запись блокируется, когда выполняется команда updatebatch (пакетное изменение)

Процедурное программирование в DAO

Название текущей версии DAO — **MS DAO 3.6 Object Library**. Обращение к элементам библиотек классов ***DAO*** и ***ADO*** и все обращения в запросах ***SQL:***
имяКласса**.** имяОбъекта**.** Элемент
Составные имена заключаются в скобки [].

1. класс Workspace - доступ к данным модели ***DAO***:
DBEngine!CreateWorkspace (имяРабочейОбласти, Пользователь, Пароль [,Тип])

2. метод CreateDatabase - создание новых БД:
[рабочаяОбласть].CreateDatabase (имяБазыДанных,Язык, [Параметры])
где *рабочаяОбласть* - ссылка на объект *Workspace*, *Язык (*константы *dbLangGeneral*, *dbLangCirillic)* определяет порядок сортировки данных, а необязательные *Параметры* задают формат ядра Jet и необходимость шифрования.

3. метод OpenDatabase - открытие БД:
[базаДанных.][рабочаяОбласть.]**OpenDatabase**(имяБазыДанных[,Монопольность [,толькоЧтение[,Источник]]])

Если БД уже открыта, к ней удобно обращаться через функцию *CurrentDb*:
CurrentDB!имяТаблицы!имяПоля.имяСвойства , CurrentDB!QueryDefs! имяЗапроса

Результирующие множества записей – объекты классов *TableDef* и *QueryDef* – создаются методами CreateTableDef, CreateQueryDef:

[рабочаяОбласть.]базаДанных.CreateTableDef («имяТаблицы»)

[рабочаяОбласть.]базаДанных.CreateQueryDef ([запрос SQL])

Объекты классов *TableDef* и *QueryDef* открываются методом OpenRecordset и закрываются с удалением из семейства *Databases* методом Close объекта *Database:*

- базаДанных.OpenRecordset (Источник [,Тип, Параметры])
- объект.OpenRecordset (Источник [,Тип, Параметры])
- базаДанных.Close
- объект.Close

Поля таблиц вначале создаются, затем добавляются в семейства, после чего обновляется окно БД:

Set объектПоле = объектТаблица.CreateField (имяПоля, Тип, [Размер])

объектТаблица.Fields.Append объектПоле

базаДанных.TableDefs.Append объектТаблица

RefreshDatabaseWindow

Доступ к полям реализуется через объекты класса *Fields*:

- имяТаблицыИлиЗапроса.Fields!имяПоля
- имяТаблицыИлиЗапроса. имяПоля
- имяТаблицыИлиЗапроса!Fields!имяПоля
- имяТаблицыИлиЗапроса! имяПоля
- Parent!имяПоляГлавнойФормы

Извлечение информации с помощью **DAO**:

1. создать рабочую область (объект Workspace)
2. открыть БД (объект Database)
3. создать набор записей (объект Recordset), выбрать записи и поля

Рабочая область - метод *CreateWorkspace* объекта *DBEngine*:

```
Set РабочаяОбласть = CreateWorkspace(Name, UserName, Password, UseType)
```

Рабочая области MS Jet:

```
Dim РабочаяОбласть As CreateWorkspace
Set РабочаяОбласть = DBEngine.CreateWorkspace(Name:="МояОбласть", _
UserName:="admin", Password:="", UseType:=dbUseJet)
```

Рабочая область ODBCDirect (ссылка на объект DBEngine применяется по умолчанию, поэтому во второй инструкции объект DBEngine опущен):

```
Dim РабочаяОбласть As CreateWorkspace
Set РабочаяОбласть = CreateWorkspace(Name:="МояОбласть", UserName:="UID",
Passwords:="", UseType:=dbUseODBC)
```

Открыть БД можно методом OpenDatabase объекта Workspace.

```
Set БазаДанных = РабочаяОбласть.OpenDatabase (name, options, readonly, connect)
```

Объект DAO Recordset

В модели **DAO** присутствуют 4 типа объектов *RecordSet*:

1. Тип Table представляет набор записей одной таблицы открытого файла БД. Он не обрабатывает связанные таблицы и таблицы ODBC и обслуживает только рабочие области Jet.
2. Тип Dynaset представляет динамический набор записей таблицы открытой БД, связанной таблицы, результата выполнения запроса или оператора SELECT. Он состоит из ссылок, поэтому обрабатывается медленнее, чем *Table* и иногда не обновляется, но охватывает более широкую область данных.
3. Тип Snapshot представляет статическую копию таблицы, запроса или оператора SQL SELECT, удобную для выборки данных и создания отчетов.
4. Тип Forward-Only представляет аналогичную копию, предназначенную для единовременного просмотра данных.

БД.OpenRecordSet (Источник [, Тип, Параметры, Блокировка])

Здесь *Источник* – это строка с именем таблицы, запроса или текстом SQL, далее следует тип объекта *RecordSet,* по умолчанию *Table* для таблиц и *Dynaset* для запросов и связанных таблиц. Любой объект *RecordSet* существует только в рамках своей процедуры, а затем уничтожается. Его можно закрыть раньше методом *Close*.

Используйте объект Recordset для:

- создания вложенных форм, т.е. формы с подчиненной формой, для обращения к одному и тому же множеству данных. Это позволит синхронизировать представление данных

  Set Me.Recordset = Forms!Form1.Recordset
- обращения к методам объекта Recordset**,** не поддерживаемым свойствами формы. Например, с методом Find для поиска записи по параметрам, заданным в полях формы.

```
Sub РабочаяОбластьDAO ()
    Set область = DBEngine.Workspaces (0)
    Set новаяБД = область.CreateDatabase ("Школьники.mdb", dbLangGeneral)
    Set другаяБД = область.OpenDatabase ("Транспорт")
    MsgBox другаяБД.Name & Chr (13) & новаяБД.Name & Chr (13) & CurrentDb.Name
End Sub
```

В отличие от режима работы через интерфейс пользователя, ***VBA*** может одновременно открыть несколько БД, хотя на экране отображается только одна из них. Полный путь к открытой БД возвращает выражение **объект.Name.**

```
Sub БазыДанныхDAO()
    Set текущаяБД= CurrentDb()
    Set новаяБД = CreateDatabase ("Студенты.mdb", dbLangCyrillic)
    Set другаяБД = OpenDatabase ("Пользователи.mdb")
    MsgBox другаяБД.Name & Chr (13) & новаяБД.Name & Chr (13) & текущаяБД.Name
End Sub

Sub НаборыЗаписейDAO ()
    Set tdf1 = CurrentDb.OpenRecordset ("Сотрудники", dbOpenDynaset)
    Set tdf2 = CurrentDb.OpenRecordset ("Категории")
    Set tdf3 = CurrentDb.OpenRecordset ("SELECT * FROM Поставщики")
    Set tdf4 = CurrentDb!Поставщики.OpenRecordset
    Set tdf5 = CurrentDb.QueryDefs (“Продажи“).OpenRecordset
    tdf.Close
End Sub
```

```
Sub УправлениеТаблицей ()
        Set tdf = CurrentDb.OpenRecordset ("Отделы")
                MsgBox "Таблица открыта"
        tdf.Close
                MsgBox "Таблица закрыта"
End Sub

Sub СозданиеТаблицы ()
        Set область = DBEngine.Workspaces (0)
        Set БД = область.OpenDatabase (“C:\Учащиеся.mdb“)
        Set таблица = БД.CreateTableDef ("НоваяТаблица")
        таблица.Fields.Append таблица.CreateField (“Кто“, dbText)
        таблица.Fields.Append таблица.CreateField (“Когда“, dbDate)
        БД.TableDefs.Append таблица
        БД.TableDefs.Refresh
        Set таблица = Nothing
        Set БД = Nothing
End Sub

Sub УдалениеТаблицы ()
        CurrentDb.TableDefs.Delete “НоваяТаблица“
        CurrentDb.TableDefs.Refresh
        RefreshDatabaseWindow
End Sub
```

```
Sub ДоступКПолямDAO ()
    MsgBox CurrentDb!Таблица1.Fields(1).Name & Space (2) & CurrentDb!
    Таблица1.Fields(2).Name & Chr (13) & "Всего полей: " & CurrentDb!
    Сотрудники.Fields.Count
End Sub

Sub ЗапросDAO ()
    s=CurrentDB.CreateQueryDef ("зСотр", "SELECT * FROM Сотрудники ORDER BY Сотрудники.ФИО")
    RefreshDatabaseWindow
End Sub
```

При создании объекта *RecordSet* строки данных помещаются в буфер и не выводятся на экран, а указатель позиционирует на текущей записи. При открытии набора записей активной становится первая запись. Для перемещения к другим записям используются методы *MoveFirst, MoveNext, MovePrevious, MoveLast.*
Методом *BookMark* можно определять закладки и возвращаться впоследствии к запомненным в них записям:

```
объектЗакладка = объектНабор.BookMark
...
объектНабор.BookMark = объектЗакладка
```

Метод *Move числоСтрок* [,*Закладка*] смещает указатель на требуемое число записей вперед или назад.

Свойства *BOF* и *EOF* объекта *Recordset* фиксируют выход за пределы набора записей.

Для поиска определенной записи в наборах типа *Table* используется метод Seek, а в наборах других типов – методы *FindFirst, FindNext, FindPrevious, FindLast*. Найденная запись становится текущей, а свойство *NoMatch* устанавливается в *False*. При отсутствии искомой записи *NoMatch = True*:

• *объект.Seek «оператор», списокКлючей*

• *объект.Find… «критерий»*

Используются операторы >, <, >=, <=, =, а список ключей описывает поля текущего индекса. Критерий представляет логическое выражение вида «Поле оператор Значение».

В модели ***DAO*** изменения в таблицы вносятся последовательно. Предварительно запись копируется в специальную область памяти – буфер копирования методом Edit, а затем методом Update возвращается в объект *Recordset*. Update сохраняет изменения. При необходимости буфер копирования очищается методом CancelUpdate:

объект.Move… или Find… или Seek
объект.Edit
объект.имяПоля = Значение
объект.Update или объект.CancelUpdate

Новые записи не добавляются прямо в БД, а сначала помещаются в буфер копирования, затем обновляют набор методом AddNew. В наборах типа *Dynaset* и *Table* без индекса новая запись добавляется в конец набора данных. Для добавленной записи автоматически создается закладка в свойстве LastModified. Указатель текущей записи при этом автоматически не перемещается.

Удаляемая запись помещается в буфер удаления и остается текущей. Удаление выполняется без предупреждения и отмены. БД Access необходимо сжимать (до 2 МБ).

```
Sub ПоискDAO ()
    Set tdf1 = CurrentDb.OpenRecordset ("Сотрудники")
    tdf1.Index = "PrimaryKey"
    tdf1.Seek "=", "Иванов"
    Set tdf2 = CurrentDb.OpenRecordset ("Командировки", dbOpenSnapshot)
    tdf2.FindFirst "Кто = 'Алексеев'"
End Sub

Sub ЗаписиDAO ()
    Set базаДанных = CurrentDb
    строкаSQL = "SELECT * FROM Таблица1 WHERE Фирма LIKE 'I*'"
    Set результатЗапроса = БД.OpenRecordset (строкаSQL)
    результатЗапроса.MoveFirst
    MsgBox результатЗапроса.Fields(0) & " - " & результатЗапроса.Fields(2)
    результатЗапроса.FindFirst "Фирма = 'Intel'"
    MsgBox результатЗапроса.Fields(0) & " - " & результатЗапроса.Fields(2)
    результатЗапроса.FindNext "Фирма = 'Intel'"
    MsgBox результатЗапроса.Fields(0) & " - " & результатЗапроса.Fields(2)
    результатЗапроса.MoveLast
    MsgBox результатЗапроса.RecordCount
End Sub
```

```
Sub ОбновлениеЗаписейDAO ()
    Set rst = CurrentDb.OpenRecordset ("Таблица1")
    где = "Фирма = 'Intel'"
    DoCmd.RunSQL "UPDATE Таблица1 SET Цена = 5 WHERE NТовара=2"
    DoCmd.RunSQL "UPDATE Таблица1 SET Фирма = 'LG' WHERE " & где
    DoCmd.RunSQL "UPDATE Таблица1 SET Фирма = 'Zilog' WHERE " & "Фирма = 'LG'"
    rst.Close
    Set rst = Nothing
End Sub

Sub АнализЗаписейDAO ()
    Set db = CurrentDb
    Set rst = db.OpenRecordset ("Таблица1", dbOpenDynaset)
    где = "Фирма = 'Intel'"
    rst.MoveFirst
    rst.FindFirst где
    If rst.NoMatch = False Then MsgBox "Фирма есть в этой таблице"
    rst.FindNext где
    If rst.NoMatch=True Then MsgBox "Фирма только одна"
    rst.Close
    db.Close
    Set rst = Nothing
    Set db = Nothing
End Sub
```

```
Sub СписокБазыДанныхDAO ()
    Set текущаяБД = CurrentDb
    Set рабочаяОбласть = DBEngine.Workspaces(0)
    Set новаяБД = рабочаяОбласть.CreateDatabase ("Студенты.mdb", dbLangGeneral)
    Set другаяБД = рабочаяОбласть.OpenDatabase ("ФирмаГодПрезидент")
    For Each БД In рабочаяОбласть.Databases
        список = список & Chr (13) & БД.Name
    Next БД
    MsgBox список
End Sub

Sub СписокПолейDAO ()
    Set db = CurrentDb
    Set tdf = db!Таблица1
    For Each fld In tdf.Fields
        строка = строка & Chr (13) & fld.Name
    Next fld
    MsgBox строка
End Sub
```

```
Sub ЗаменаДанныхDAO ()
    Set объект = CurrentDb.OpenRecordset ("Товары", dbOpenDynaset)
    критерий = "NТовара = 1"
    объект.FindFirst критерий
    Do Until объект.NoMatch
        With объект
            .Edit
            .NТовара = 333
            .Update
            .FindNext критерий
        End With
    Loop
    объект.Close
    Set объект = Nothing
End Sub

Sub ЧтениеДанныхDAO ()
‘ В Access для загрузки в массив строк объекта Recordset используется метод GetRows
    Set объект = CurrentDb.OpenRecordset("Товары")
    массив = объект.GetRows (объект.RecordCount)
    For Each элемент In массив
        строка = строка & элемент & chr (13)
    Next
    MsgBox строка
End Sub
```

```
Sub ПравкаТаблицDAO ()
    Set таблица = CurrentDB.OpenRecordset ("Таблица1")
    таблица.MoveFirst
    таблица.Edit
    таблица.Fields (Фирма) = "SCAN"
    таблица.Fields (Год) = 2005
    строкаSQL = "UPDATE Таблица1 SET NTовара = 22 WHERE NTовара = 2 "
    CurrentDb.Execute строкаSQL
    таблица.AddNew
    таблица.Fields (ФИО) = “Новый“
    таблица.Update
    таблица.BookMark =таблица.LastModified
    таблица.Move 5
    таблица.Delete
    таблица.moveNext
End Sub
```

Создание таблицы с помощью DAO

```
Option Explicit                    'предписание явного объявления переменных
Sub Create_table()
Dim db As Database, td As TableDef, fld As Field ‘Объектные переменные для БД, таблиц и полей
Set db = CurrentDb                  ‘Ссылка на текущую БД
Set td = db.CreateTableDef("Временная")    ‘Новая таблица, метод CreateTableDef объекта Database
Set fld = td.CreateField("НомерЗачетки", dbByte) ‘Тип поля Byte, метод CreateField объекта TableDef
td.Fields.Append fld                'Добавляем поле  в семейство Fields таблицы
Set fld = td.CreateField("Фамилия", dbText) ‘Тип поля текстовый
td.Fields.Append fld                ‘Добавляем поле "Фамилия" в семейство Fields таблицы
db.TableDefs.Append td              'Добавляем таблицу к семейству TableDefs БД
db.TableDefs.Refresh                'Обновляем количество объектов семейства TableDefs
End Sub
```

Удаление таблицы с помощью DAO

```
Sub Del_table()
Dim db As Database
Set db = CurrentDb                  'Устанавливаем ссылку на текущую БД
db.TableDefs.Delete "Временная"
db.TableDefs.Refresh
Set db = Nothing                    'Освобождаем объектную переменную
End Sub
```

Создание запроса с помощью DAO

```
Public Sub CreateQueryDAO()
Dim db As Database, qd As QueryDef, rs As DAO.Recordset
Set db = CurrentDb
Set qd = db.CreateQueryDef("Отличники") 'новый запрос сохраняем в БД
qd.SQL = "SELECT код, предмет, оценка FROM Экзамены WHERE (оценка)=5"
Set rs = qd.OpenRecordset(dbOpenDynaset) ‘набор записей на базе нового запроса
Set rs = Nothing
End Sub
```

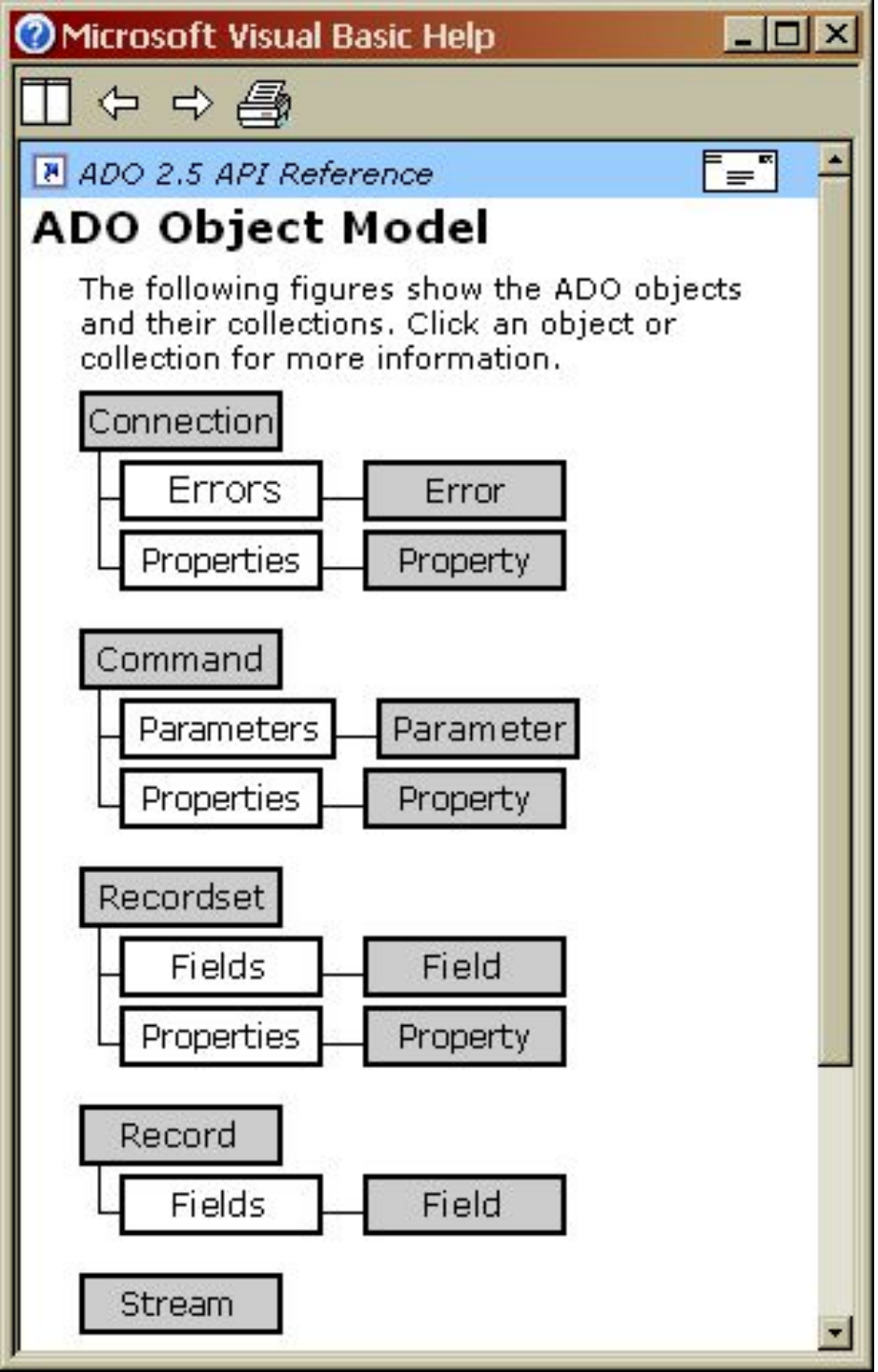
```
Sub ConnectionObjectX()
Dim wrkJet as Workspace, dbsNorthwind As Database, wrkODBC As Workspace, conPubs As
Connection
Set wrkJet = CreateWorkspace("NewJetWorkspace", "admin", "", dbUseJet)      ' (DAO)
Set dbsNorthwind = wrkJet.OpenDatabase("Northwind.mdb")
Set wrkODBC = CreateWorkspace("NewODBCWorkspace", "admin", "", dbUseODBC) ‘(DAO)

‘The DSNs must be configured to use Microsoft Windows NT Authentication Mode to authorize
user access to the MS SQL Server
Set conPubs=wrkODBC.OpenConnection("Connection1", , , ODBC; DATABASE=pubs;
DSN=Publishers")
Debug.Print "Database properties:"
With dbsNorthwind
      For Each prpLoop In .Properties
      Debug.Print "  " & prpLoop.Name & " = " & prpLoop.Value
      Next prpLoop
End With
For Each conLoop In wrkODBC.Connections
      Debug.Print "Connection properties for " & conLoop.Name & ":"
Next conLoop
dbsNorthwind.Close
conPubs.Close
wrkJet.Close
wrkODBC.Close
End Sub
```

Строка соединения — это единое целое, включающее в себя блоки информации о провайдере и пути к БД, разделенные символом точки с запятой

Объекты ADO

- Connection
- Recordset



```
Dim conn As ADODB.Connection              'declare conn to be a Connection
Set conn = New ADODB.Connection           ' make a connection object
conn.Provider = "Microsoft.Jet.OLEDB.4.0“      ' specify what kind of data provider it is
conn.Open "c:/walter/ass21.mdb“           ' open the connection on one database
Dim myTableRS As ADODB.Recordset          ' declare a recordset
Set myTableRS = New ADODB.Recordset       ' make one
' open it using a table in the database, and the connection
myTableRS.Open "myTable", conn, adOpenDynamic, adLockPessimistic
myTableRS.MoveFirst                       ' go to start of recordset
' until we reach the end..
Do Until myTableRS.EOF
      MsgBox (myTableRS.Fields("ID"))     ' display the ID field in current row
      myTableRS.MoveNext                  ' move next row
Loop
myTableRS.Close                           'close the recordset
Set myTableRS.ActiveConnection = Nothing
conn.Close                                ' and the connection
Set conn = Nothing
```

Поиск записи

```
Dim conn As ADODB.Connection
Dim myTableRS As ADODB.Recordset
Set conn = New ADODB.Connection
Set myTableRS = New ADODB.Recordset
conn.Provider = "Microsoft.Jet.OLEDB.4.0"
conn.Open "c:/walter/ass21.mdb"
myTableRS.Open "myTable", conn, adOpenStatic, adLockOptimistic
```

```
Dim wanted As String
Text5.SetFocus
wanted = Text5.Text

myTableRS.Find "ID = " & wanted
If Not myTableRS.EOF Then
  Label8.Caption = myTableRS.Fields("Name")
Else
  Label8.Caption = "Not found"
End If
```

Поиск записиFind a row with a certain key field value and display other field

Выбор значения их текстового поля

Поиск

Отображение результатов

Запись результатов в БД

```
Dim conn As ADODB.Connection
Dim myTableRS As ADODB.Recordset
Set conn = New ADODB.Connection
Set myTableRS = New ADODB.Recordset
conn.Provider = "Microsoft.Jet.OLEDB.4.0"
conn.Open "c:/bd1.mdb"
myTableRS.Open "myTable", conn, adOpenStatic, adLockOptimistic
myTableRS.MoveFirst
Do While Not myTableRS.EOF
    myTableRS.Fields("PhoneNumber") =
    myTableRS.Fields("PhoneNumber") + 1
    myTableRS.Update
    myTableRS.MoveNext
Loop

myTableRS.Close
Set myTableRS.ActiveConnection = Nothing
conn.Close
```

Вставка записей

```
myTableRS.Open "myTable", conn, adOpenDynamic, adLockPessimistic
myTableRS.AddNew
nameTxtBox.SetFocus
myTableRS.Fields("Name") = nameTxtBox.Text
phoneTxtBox.SetFocus
myTableRS.Fields("PhoneNumber") = phoneTxtBox.Text
myTableRS.Update
myTableRS.Close
```

Новая запись добавляется в конец таблицы. В РБД порядок записей не имеет значения.

Удаление записей

```
IDTxtBox.SetFocus
myTableRS.Find "ID = " & IDTxtBox.Text
If Not myTableRS.EOF Then
    myTableRS.Delete
    myTableRS.Update
    MsgBox ("Record deleted")
Else
    MsgBox ("No matching record")
End If
myTableRS.Close
```

```
myTableRS.Open "Select ID, name From myTable", conn, adOpenDynamic,
adLockPessimistic
Do While Not myTableRS.EOF
For i = 1 To myTableRS.Fields.Count
Debug.Print myTableRS.Fields(i - 1),
Next
Debug.Print
myTableRS.MoveNext
Loop
```

Использование SQL

```
Dim conn As ADODB.Connection
Set conn = New ADODB.Connection
conn.Provider = "Microsoft.Jet.OLEDB.4.0"
conn.Open "c:/db1.mdb"
Dim myCommand As ADODB.command
Set myCommand = New ADODB.command
myCommand.ActiveConnection = conn
myCommand.CommandText="Update myTable set phone=phone + 2"
myCommand.Execute
conn.Close
Set conn = Nothing
```

Объект Command

Создание таблицы с помощью ADO

```
Sub ADO()
Dim cnn As New ADODB.Connection          'Соединение с текущей БД
Dim cat As New ADOX.Catalog
Set cnn = CurrentProject.Connection          'Используется объект модели объектов Access
cat.ActiveConnection = cnn
Debug.Print cat.Tables(0).Type
Dim Table
Set Table = CreateObject("ADOX.Table")          'Создаем таблицу в ADO
Table.name = "Временная_2"
Table.Columns.Append "НомерЗачетки", dbInteger      'Создаем столбец
Table.Columns.Append "Фамилия", dbText
cat.Tables.Append Table                  'Добавляем таблицу к семейству Tables
Set cat = Nothing
End Sub
```

Удаление таблицы с помощью ADO

```
Sub ADO_del()
Dim cnn As New ADODB.Connection              'Соединение с текущей БД
Dim cat As New ADOX.Catalog
Set cnn = CurrentProject.Connection          'Используется объект модели объектов Access
cat.ActiveConnection = cnn
Debug.Print cat.Tables(0).Type
cat.Tables.Delete ("Временная_2")
End Sub
```

Редактирование таблиц с помощью DAO и ADO

```
Sub ADO_1()
Dim Cnn As New ADODB.Connection, rsADO As New ADODB. Recordset
Set cnn = New ADODB.Connection
Set rsADO = New ADODB.Recordset
Cnn.Mode=adModeShareDenyNone    ‘по умолчанию Shared
cnn.Open "Provider=Microsoft.jet.oledb.4.0;Data Source=D:/Wind.mdb;”
‘ Set con = CurrentProject.Connection
rsADO.Open "tblAccount", cnn, adOpenKeyset, adLockOptimistic
Do While Not rsADO.EOF
  rsADO![Commis]=trim(rsADO![Commis])
  rsADO.Update
  rsADO.MoveNext
Loop
Set rsADO = Nothing
Set cnn = Nothing
rsADO.Close
cnn.Close
End Sub
```

```
Sub DAO_1()
Dim db As DAO.Database, Dim rsDAO As DAO.Recordset
Set db = DBEngine.OpenDatabase(“D:/Wind.mdb“, ReadOnly:=True, Shared:=True)
‘ Set db = CurrentDb()
Set rsDAO = db.OpenRecordset("tblAccount")
Do While Not rsDAO.EOF
  rsDAO.Edit
  rsDAO![Commis]=trim(rsDAO![Commis])
  rsDAO.Update
  rsDAO.MoveNext
Loop
Set rsDAO = Nothing
Set db = Nothing
rsDAO.Close
cnn.Close
End Sub
```

Functionality	DAO	ADO[1]	ADOX[2]	JRO (*)
Create Recordsets	X	X		
Edit Startup properties	X	X**		
Support ANSI-92 SQL		X	X	
Create Tables	X		X	
Create New Database	X		X*	
Edit Existing Table properties	X		X	
Create table relationships	X		X*	
Create New Users/Groups	X		X	
Edit security settings	X		X*	
Support for new Jet 4.0 Decimal datatype			X	
Support for Compression attribute for column data			X	
Edit stored, basic SQL queries or views	X		X*	
Create permanent queries that are accessible only through code			X*	
Create queries accessible through database container/UI and code	X			
Compact/Encode database	X			X[4]
Refresh Cache	X			X
Make Database Replicable	X			X[3]
Make Database Replicas	X			X[3]
Synchronize Replicas	X			X[3]
Edit Database properties	X			
Create custom database properties	X			
Edit table column properties	X			

* - only .mdb, ** - only .adp, 1 - Uses Connection object to reference to database; 2 - Uses Catalog object to reference database; 3 - Uses Replica object to reference database; 4 - Uses JetEngine object to reference database

DAO	ADO (ADODB)
DBEngine	None
Workspace	None
Database	Connection
Recordset	Recordset
Dynaset-Type	Keyset
Snapshot-Type	Static
Table-Type	Keyset с параметром adCmdTableDirect
Field	Field
Конвертация кода DAO в код ADO	**Open a Recordset**
Dim db as Database Dim rs as DAO.Recordset Set db = CurrentDB() Set rs=db.OpenRecordset("Employees")	Dim rs as New ADODB.Recordset rs.Open "Employees", CurrentProject.Connection, adOpenKeySet, adLockOptimistic
	Edit a Recordset
rs.Edit rs("TextFieldName") = "NewValue" rs.Update	rs("TextFieldName") = “NewValue” rs.Update Перенос фокуса с текущей записи методами MoveNext, MoveLast, MoveFirst, MovePrevious без метода CancelUpdate приведет к выполнению метода

DAO в ADO с помощью Microsoft Jet Provider

DAO

```
Sub DAOOpenJetDatabase()
    Dim db As DAO.Database
    Set db = DBEngine.OpenDatabase(".\NorthWind.mdb")
    db.Close
End Sub
```

ADO

```
Sub ADOOpenJetDatabase()
    Dim cnn As New ADODB.Connection
    cnn.Open "Provider=Microsoft.Jet.OLEDB.4.0; Data Source=.\NorthWind.mdb;"
    cnn.Close
End Sub
```

DAO

```
Sub DAOOpenJetDatabaseReadOnly()
    Dim db As DAO.Database   ' Open shared, read-only.
    Set db = DBEngine.OpenDatabase (".\NorthWind.mdb", False, True)
    db.Close
End Sub
```

ADO

```
Sub ADOOpenJetDatabaseReadOnly()
    Dim cnn As New ADODB.Connection   ' Open shared, read-only
    cnn.Mode = adModeRead
    cnn.Open "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=.\NorthWind.mdb;"
    cnn.Close
End Sub
```

DAO

```
Sub DAOOpenRecordset()
Dim db As DAO.Database, rst As DAO.Recordset, fld As DAO.Field
Set db = DBEngine.OpenDatabase(".\NorthWind.mdb")
' Open the forward-only, read-only recordset
Set rst = db.OpenRecordset ("SELECT * FROM Customers WHERE Region = 'WA'",
dbOpenForwardOnly, dbReadOnly)
' Print the values for the fields in the first record in the debug window
  For Each fld In rst.Fields
     Debug.Print fld.Value & ";";
  Next
rst.Close
End Sub
```

ADO

```
Sub ADOOpenRecordset()
Dim cnn As New ADODB.Connection, rst As New ADODB.Recordset, fld As ADODB.Field
cnn.Open "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=.\NorthWind.mdb;"
' Open the forward-only, read-only recordset
rst.Open "SELECT * FROM Customers WHERE Region = 'WA'", cnn, adOpenForwardOnly,
adLockReadOnly
 For Each fld In rst.Fields
     Debug.Print fld.Value & ";";
  Next
rst.Close
End Sub
```

```
DAO        ' Open the forward-only, read-only recordset
Sub DAOMoveNext()
Dim db As DAO.Database, rst As DAO.Recordset, fld As DAO.Field
Set db = DBEngine.OpenDatabase (".\NorthWind.mdb")
Set rst = db.OpenRecordset("SELECT * FROM Customers WHERE Region = 'WA'",  dbOpenForwardOnly,
dbReadOnly)
While Not rst.EOF
   For Each fld In rst.Fields
     Debug.Print fld.Value & ";";
   Next
   rst.MoveNext
Wend
rst.Close
End Sub
ADO        ' Open the forward-only, read-only recordset
Sub ADOMoveNext()
Dim cnn As New ADODB.Connection, rst As New ADODB.Recordset, fld As ADODB.Field
cnn.Open "Provider=Microsoft.Jet.OLEDB.4.0; Data Source=.\NorthWind.mdb;"
rst.Open "SELECT * FROM Customers WHERE Region = 'WA'", cnn, adOpenForwardOnly,
adLockReadOnly
While Not rst.EOF
   For Each fld In rst.Fields
     Debug.Print fld.Value & ";";
   Next
   rst.MoveNext
Wend
rst.Close
End Sub
```

DAO

```
Sub DAOAddRecord()
Dim db As DAO.Database, rst As DAO.Recordset
Set db = DBEngine.OpenDatabase (".\NorthWind.mdb")
Set rst = db.OpenRecordset("SELECT * FROM Customers", dbOpenDynaset)
rst.AddNew
' Specify the values for the fields
rst!CustomerId = "HENRY"
rst.Update     ' Save the changes you made to the current record in the Recordset
' Position recordset on new record
rst.Bookmark = rst.LastModified
Debug.Print rst!CustomerId
rst.Close
End Sub
```

В **DAO** запись, которая была текущей перед вставкой новой записи, остается текущей записью, поэтому переводим курсор на новую запись.

В **ADO** новая запись сразу становится текущей.

ADO

```
Sub ADOAddRecord()
Dim cnn As New ADODB.Connection, rst As New ADODB.Recordset
cnn.Open "Provider=Microsoft.Jet.OLEDB.4.0; Data Source=.\NorthWind.mdb;"
rst.Open "SELECT * FROM Customers", cnn, adOpenKeyset, adLockOptimistic
rst.AddNew
' Specify the values for the fields
rst!CustomerId = "HENRY"
rst.Update
Debug.Print rst!CustomerId
rst.Close
End Sub
```

DAO

```
Sub DAOUpdateRecord()
Dim db As DAO.Database, rst As DAO.Recordset
Set db = DBEngine.OpenDatabase(".\NorthWind.mdb")
Set rst = db.OpenRecordset("SELECT * FROM Customers WHERE CustomerId = 'LAZYK'",
dbOpenDynaset)
rst.Edit
' Update the Contact name of the first record
  rst.Fields("ContactName").Value = "New Name"
rst.Update
rst.Close
End Sub
```

ADO

```
Sub ADOUpdateRecord()
Dim cnn As New ADODB.Connection, rst As New ADODB.Recordset
cnn.Open "Provider=Microsoft.Jet.OLEDB.4.0; Data Source=.\NorthWind.mdb;"
rst.Open "SELECT * FROM Customers WHERE CustomerId = 'LAZYK'", cnn, adOpenKeyset,
adLockOptimistic
rst.Fields("ContactName").Value ="New Name"
rst.Update
rst.Close
End Sub
```

DAO ‘Executing a nonparameterized stored query

```
Sub DAOExecuteQuery()
  If gbBreakEach Then Stop
  Dim db As DAO.Database, rst As DAO.Recordset, fld As DAO.Field
Set db = DBEngine.OpenDatabase(".\NorthWind.mdb")
Set rst = db.OpenRecordset("Products Above Average Price",  dbOpenForwardOnly, dbReadOnly)
While Not rst.EOF
    For Each fld In rst.Fields
      Debug.Print fld.Value & ";";
    Next
    rst.MoveNext
  Wend
rst.Close
End Sub
```

ADO

```
Sub ADOExecuteQuery()
  Dim cnn As New ADODB.Connection, rst As New ADODB.Recordset, fld As ADODB.Field
cnn.Open "Provider=Microsoft.Jet.OLEDB.4.0; Data Source=.\NorthWind.mdb;"
rst.Open "[Products Above Average Price]", cnn, adOpenForwardOnly,  adLockReadOnly,
adCmdStoredProc
While Not rst.EOF
    For Each fld In rst.Fields
      Debug.Print fld.Value & ";";
    Next
    rst.MoveNext
  Wend
rst.Close
End Sub
```

DAO

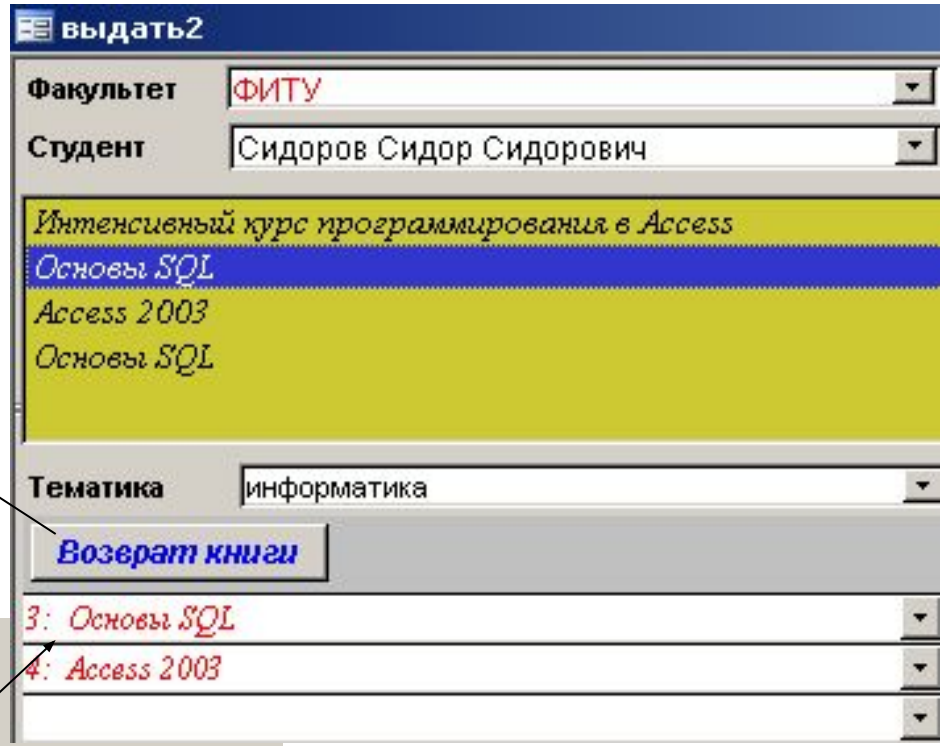
```
'Executing a parameterized stored query
Sub DAOExecuteParamQuery()
Dim db As DAO.Database, qdf As DAO.QueryDef, rst As DAO.Recordset, fld As DAO.Field
Set db = DBEngine.OpenDatabase(".\NorthWind.mdb")
Set qdf = db.QueryDefs("Sales by Year") ' Get the QueryDef from the QueryDefs collection
qdf.Parameters("Forms!Sales by Year Dialog!BeginningDate") = #8/1/1997#  ' Specify the parameter values
Set rst = qdf.OpenRecordset(dbOpenForwardOnly, dbReadOnly)
While Not rst.EOF
   ..............
   rst.MoveNext
 Wend
rst.Close
End Sub
```

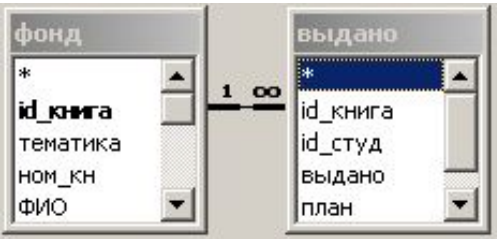
ADO

```
Sub ADOExecuteParamQuery()
Dim cnn As New ADODB.Connection,  cat As New ADOX.Catalog, cmd As ADODB.Command, rst As New
ADODB.Recordset
cnn.Open "Provider=Microsoft.Jet.OLEDB.4.0; Data Source=.\NorthWind.mdb;"
cat.ActiveConnection = cnn
Set cmd = cat.Procedures("Sales by Year").Command ' Get the Command object from the Procedure
cmd.Parameters("Forms![Sales by Year Dialog]!BeginningDate") = #8/1/1997#
rst.Open cmd, , adOpenForwardOnly,adLockReadOnly, adCmdStoredProc
While Not rst.EOF
   ..............
   rst.MoveNext
 Wend
rst.Close
End Sub
```

БД Библиотека

```
Dim rst As Recordset, strSQL as string
Screen.PreviousControl.SetFocus
If Not IsNull(Me.id_книга) Then
   If Not IsNull(Me.прим) Then
      strSQL = "SELECT [фонд].[id_книга], [фонд].[прим] “ & _
      “FROM фонд WHERE ([фонд].[id_книга]=" & Me.id_книга & ");"
      Set rst = CurrentDb.OpenRecordset(strSQL)
      rst.Edit
      If IsNull(rst![прим]) Then
         rst![прим] = Me.прим
      Else
         rst![прим] = rst![прим] & "; " & Me.прим
      End If
      rst.Update
      rst.Close
      Set rst = Nothing
   End If
End If
```



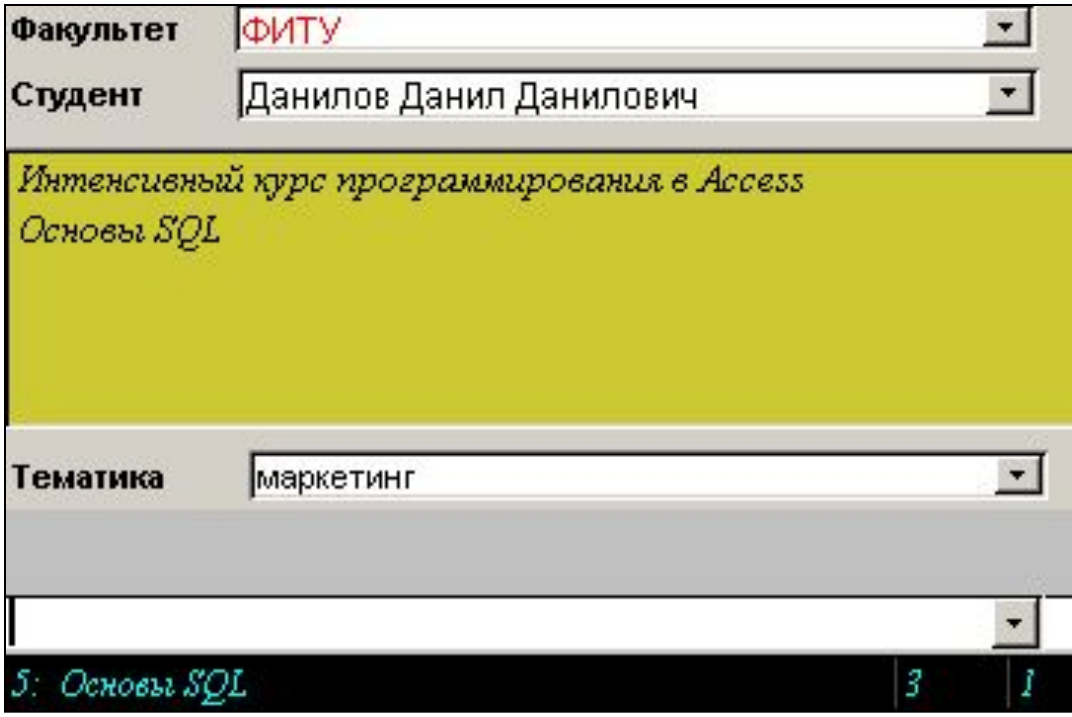


id_книга	выдано	план	прим	id_студ	тематика
выдано	выдано	выдано	выдано	выдано	фонд
☑	☑	☑	☑	☑	☐
				[Forms]![выдать2].[id_студ]	[Forms]![выдать2].[тематика]

Набор записей, отображаемый в подчиненнной форме

```
Private Sub id_книга_BeforeUpdate(Cancel As Integer)
Dim rst1, rst2 As Recordset,  strSQL1 as String, strSQL2 As String
If Me.id_книга <> 0 Then
strSQL1="SELECT фонд.кол FROM фонд WHERE (фонд.id_книга=" & Me.id_книга & "); "
Set rst1=CurrentDb.OpenRecordset(strSQL1)
strSQL2= "SELECT count(выдано.id_книга) as count1 FROM выдано INNER JOIN фонд ON
выдано.id_книга = фонд.id_книга WHERE (фонд.id_книга=" & Me.id_книга & "); "
    Set rst2 = CurrentDb.OpenRecordset(strSQL2)
    If (rst1![кол] - rst2!count1) <= 0 Then
        MsgBox "Все книги на руках"
        Me.Undo
        Me![id_книга].Requery
    End If
End If
End Sub
```

Код на событии ПОСЛЕ ОБНОВЛЕНИЯ поля СПИСОК КНИГ по указанной тематике

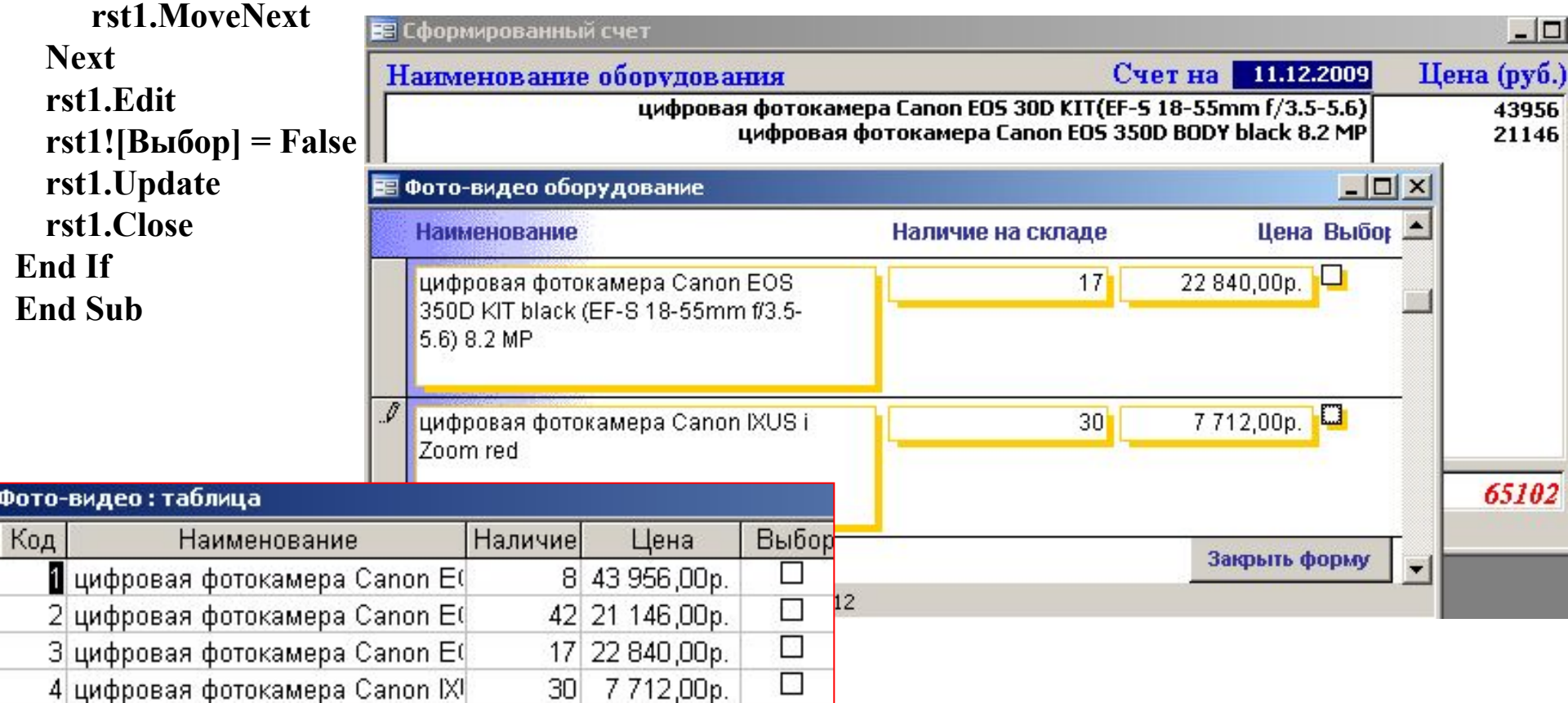


БД “Компьютерный Магазин”

```
Sub update_table(table1)
Dim strsql1 As String, rst1 As Recordset
strsql1 = "SELECT " + table1 + ".выбор FROM " + table1 + " where " + table1 + ".выбор=true"
Set rst1 = CurrentDb.OpenRecordset(strsql1, dbOpenDynaset)
If rst1.RecordCount <> 0 Then
   rst1.MoveLast
   rst1.MoveFirst
   For i = 1 To rst1.RecordCount - 1
      rst1.Edit
      rst1![Выбор] = False
      rst1.Update
      rst1.MoveNext
   Next
   rst1.Edit
   rst1![Выбор] = False
   rst1.Update
   rst1.Close
End If
End Sub
```

Код на закрытии формы Фото-видео (для снятия отметок о выборе товара в исходной таблице)

```
Private Sub Form_Unload(Cancel As Integer)
      Call update_table("[Фото-видео]")
End Sub
```



```
Dim conn As ADODB.Connection         'declare conn to be a Connection
Set conn = New ADODB.Connection      'make a connection object
conn.Provider = "Microsoft.Jet.OLEDB.4.0“'specify what kind of data provider it is
conn.Open "c:/walter/ass21.mdb“          'open the connection on one database

Dim myTableRS As ADODB.Recordset 'declare a recordset
Set myTableRS = New ADODB.Recordset    'make one

' open it using a table in the database, and the connection
myTableRS.Open "myTable", conn, adOpenDynamic, adLockPessimistic
myTableRS.MoveFirst                  'go to start of recordset
Do Until myTableRS.EOF               'until we reach the end
      MsgBox (myTableRS.Fields("ID"))'display the ID field in current row
      myTableRS.MoveNext             'move next row
Loop

myTableRS.Close                   'close the recordset
Set myTableRS.ActiveConnection = Nothing
conn.Close                        'close the connection
Set conn = Nothing
```

Протокол OLE DB

Пример использования объекта ADODB.Connection

```
Sub DemoADODB()
Const Provider = "Provider=Microsoft.Jet.OLEDB.4.0;"
Const DataSource = "Data Source=C:\Data\Hours.mdb"
Dim Connection As New ADODB.Connection
On Error GoTo Finally
Call Connection.Open(Provider & DataSource)
Connection.Close
Finally:
    If (Err.Number <> 0) Then
        MsgBox Err.Description
    End If
Set Connection = Nothing
End Sub

Sub ТекущаяЗаписьADO ()
    Dim rst As New ADODB.Recordset
    rst.Open "Сотрудники", CurrentProject.Connection
    Debug.Print rst.Fields ("Кто"), rst.Fields ("Когда")
    rst.Close
End Sub
```

Извлечение информации с помощью **ADO:**

1. Объявить переменную-объект класса ADODB.Connection.
2. Задать информацию о провайдере OLE DB
3. Открыть соединение.
4. По завершении работы освободить память, выделенную объекту Connection.

```
Sub DemoADODB()
    Const Provider = "Provider=Microsoft.Jet.OLEDB.4.0;"
    Const DataSource = "Data Source=C:\Data\Hours.mdb"
    Dim Connection As New ADODB.Connection
    On Error GoTo Finally
    Call Connection.Open(Provider & DataSource)
    …..
    Set Connection =Nothing
    Connection.Close
  Finally:
    If (Err.Number <> 0) Then
     MsgBox Err.Description
    End If
    Set Connection =Nothing
    Connection.Close
End Sub
```

После объявления переменной Recordset объект надо заполнить, используя:

- метод Open объекта Recordset;
- метод Execute объекта Command;
- метод Execute объекта Connection.

1. Метод *Open*, если используется простая инструкция Select:

```
Dim con As New Connection
Dim rst As Recordset
Dim strSQL As String
……..’ соединение и открытие БД
strSQL = "SELECT * FROM Toys"
Set rst.ActiveConnection = con
rst.Open strSQL,,adOpenForwardOnly, adLockReadOnly, adCmdText ‘нет 2-го аргумента
```

2. В приложениях клиент/сервер обращение к сохраненным запросам методом *Execute*:

```
Dim con As New Connection
Dim cmd As New Command
Dim rst As Recordset
Dim strSQL As String
……..’ соединение и открытие БД
With cmd
   Set .ActiveConnection = con
  .CommandText = strSQL
  .CommandType = adCmdText
End With
rst.CursorType = adOpenForwardOnly
rst.lockType = adLockReadOnly
Set rst = cmd.Execute()
```

3. Метода *Execute* объекта *Connection* позволяет работать с сохраненными процедурами. Однако, если для выполнения процедур необходимы определенные параметры, придется включить эти параметры в инструкцию SQL:

```
Dim conVert As New Connection
Dim rst As Recordset
Dim strSQL As String
……..' соединение и открытие БД
rst.CursorType = adOpenForwardOnly
rst.LockType = adLockReadOnly
Set rst = conVert.Execute()
```

Создание подключения на лету

Если известно, что подключение необходимо только для одного объекта Recordset, можно указать строку подключения в качестве 2-го аргумента метода Open объекта Recordset без выделения переменной для подключения к БД:

```
Dim rst As New Recordset
Dim strSQL As String, strConnect As String
strConnect = "Provider=Microsoft.Jet.OLEDB.4.0; Data Source=C:\Data\Shop.mdb"
strSQL = "SELECT * FROM Bicycles"
rst.Open strSQL, strConnect, adOpenForwardOnly
```

1. Если используется обращение к БД Jet, то Access автоматически создает объект Connection для объекта CurrentProject, который можно использовать:

```
Dim conADOConnection As Connection          ‘ ADO
Set conADOConnection = CurrentProject.Connection
```

2. Подключение с БД SQL Server в проекте Access - свойство BaseConnectionString объекта CurrentProject:

```
Dim conADO As New Connection
conADO.ConnectionString = CurrentProject.BaseConnectionString
```

3. Эквивалентные примеры создания объекта Connection для БД Jet:

```
1. Dim. conADOConnection As New Connection       ‘ ADO
   Dim strCont As String
   strCon = "Provider=Microsoft.Jet.OLEDB.4.0; Data Source=C:\Data\Toys.mdb"
   conADOConnection.Open strCon
2. Dim conADOConnection As New Connection       ‘ ADO
   With conADOConnection
         .Provider="Microsoft.Jet.OLEDB.4.0;"
         .Properties("Data Source") - "=C:\Data\Toys.mdb"
         .Open
   End With
```

4. Пример для SQL Server:

```
Dim conADOConnection As New Connection    ‘ ADO
Dim strCon As String
strCon="Provider=SQLOLEDB; Data Source=DSN1; Initial Catalog=toys;User ID=sa; Password=;"
conADOConnection.Open strCon
```

Выбор типа курсора

Курсор необходим для перемещения между записями. По умолчанию тип Forward-only.

Тип курсора	Константа-значение свойства CursorType	Определение
Forward-only	adOpenForwardOnly	Разрешает перемещение между элементами объекта только в одном направлении, или на определенное количество записей, или к последней записи. Изменения, внесенные другими пользователями, не появляются до тех пор, пока набор записей не будет закрыт, а затем снова открыт. Этот тип курсора обеспечивает максимальное быстродействие, но только в том случае, если вам необходимо перемещаться по записям всего один раз
Static	adOpenStatic	Фиксированный набор записей, который не может быть обновлен и не отражает изменения, внесенные другими пользователями, до тех пор, пока он не будет закрыт, а затем снова открыт. Этот вариант подходит для поиска данных и создания отчетов, обеспечивая более высокую скорость, чем варианты Keyset и Dynamic
Keyset	adOpenKeyset	Набор, чьи записи их значения могут изменяться в результате внесения соответствующих изменений в базу данных. Однако подобный набор записей не отражает изменения, внесенные другими пользователями
Dynamic	adOpenDynamic	Этот вариант напоминает набор записей Keyset, за исключением того, что он не отражает изменения, внесенные в базу данных

Блокирование

Свойство LockType по умолчанию имеет свойства adLockPessimistic.

Тип блокировки	Константа-значение свойства LockType	Определение
No lock (Блокировка отсутствует)	adLockOptimistic	Несколько пользователей могут одновременно изменять одну и ту же запись, однако при этом получают уведомление о том, что внесенные ими изменения могут конфликтовать с изменениями, внесенными другими пользователями (записи блокируются только при выполнении метода Update). Этот вариант хорошо подходит для баз данных, с которыми работает только один пользователь, а также многопользовательских баз данных, в которых всегда правильными и полными считаются сведения, добавленные самыми последними
Batch update (Пакетное обновление)	adLockBatchOptimistic	Этот вариант напоминает предыдущий, за тем исключением, что записи блокируются при групповом, а не индивидуальном обновлении
Read only (all records) (Только чтение (все записи))	adLockReadOnly	Все записи блокируются в то время, когда набор записей открыт; изменять их не может ни один пользователь, Однако доступ к записям разрешен
Read only (edited records) (Только чтение (изменяемые записи))	adLockPessimistic	Блокируется только та запись, с которой пользователь работает в данный момент. Блокировка снимается в тот момент, когда пользователь переходит к другой записи

Методы Recordset

- Основные
  - rst.Open
  - rst.Close
- Перемещение по записям
  - rst.Movefirst
  - rst.MoveLast
  - rst.MoveNext
  - rst.MovePrevious
- Поиск
  - rst.Find
  - rst.Seek **‘если поле проиндексировано (rst.seek ‘1234’, asSeekFirstEQ)**
- Редактирование
  - rst.AddNew
  - rst.Delete
  - rst.Update

Перемещение по набору записей (ADO)

Методы MoveFirst, MoveLast, MoveNext и MovePrevious.
Метод Move позволяет перемещаться на определенное количество записей в наборе вперед или назад. Например, инструкция rst.Move -3 перемещает на три записи назад.
Для возвращения к определенной записи, для нее создается закладка:

```
var Bookmark1 = rst.Bookmark
…
rst.Bookmark = varBookmarkl
```

Метод Seek, а также методs *FindFist, FindLat, FindNext и FindPrevious* позволяют отследить определенную запись, базируясь на ее содержимом. Метод Seek работает быстрее, однако, прежде содержимое БД должно быть проиндексировано.

Добавление и удаление записей

Метод AddNew добавляет новую запись в набор. Метод позволяет указать поля и их значения:

```
With rst
      .AddNew  Array ( "Имя", "Возраст", "Пол“),   Array("Анна", 42, "Ж")
End With
```

Для удаления текущей записи предназначен метод Delete.

Чтение поля

Работа с текущим значением - укажите поле по имени или по его индексному номеру:

```
If  rst.Fields("Service visits").Value >10
    MsgBox "This unit needs a major overhaul!"
End If
strCurrentFieldData = rst.Fields(3)
```

Поскольку *Fields* является семейством по умолчанию объекта Recordset, его указывать не обязательно:

```
rst!Date = #5/15/2001#            ‘ rst.fields!Date
With rst
     intltems =![Oil cans]   ‘ rst.fields![Oil cans]
End With
```

Изменение поля

Указать новое значение и переместиться к другой записи:

```
With rst
    .Fields(0).Value = “Lemon”
    .MoveNext
End With
```

Если надо остаться на текущей записи – метод Update:

```
With rst
    .Value=2
    .Update
End With
```

Recordset - свойства

- rst.recordCount
- rst.BOF
- rst.EOF
- rst.AbsolutePosition
- rst.Fields.Count

Обращение к полям записи

Каждая запись содержит поля, к каждому из которых можно обратиться по индексу (начиная с 0) следующими способами:

- номер индекса rec(0)
- имя поля rec(“surname”)
- переменная rec(fieldpos)
- выражение rec(fieldpos + 3)

Использование объекта Command

Объект Command представляет инструкцию SQL или сохраненную процедуру. Средства доступа OLE DB Provider не нужны. Для входных параметров определить объекты *Parameter* коллекции *Parameters*. Значение adCmdText передает инструкцию SQL источнику данных:

```
Dim con As Connection
Dim cmdVBA As Command
Dim prmDate
Set cmdVBA = New Command
With cmdVBA
    .ActiveConnection = con                    ‘ соединение определено и открыто
   .CommandText = "qryDeleteOldRecords”          ‘ имя хранимой процедуры
   .CommandType = adCmdStoredProc, adCmdTable      ' in Jet
‘    .CommandText = "UPDATE Bicycles SET OnSale = True WHERE Category = 4;"
‘    .CommandType = adCmdText
    .Execute
End With
Set prmDate =New Parameter
With prmDate
    .Name = "Date"
    .Value = InputBox "Enter the cut-off date."
    .Type = adDate
    .Direction = adParamInput
End With
With cmdVBA
    .Parameters.Append prmDate                 ' Добавление параметра
    .Execute
End With
```

Для запроса, сохраненного в БД Jet/Access, используйте значение adCmdTable свойства CommandType объекта Command, а не adCmdStoredProc, относящееся к SQL Server и другим серверам БД.

SQL

В коде VBA принято использовать одинарные кавычки для определения строки в инструкции SQL:

```
strSQL = "SELECT Name FROM Kids WHERE Hates = 'Brocolli' "
cmd.CommandText = strSQL
```

Если часть инструкции SQL основывается на переменной, например, запрос базируется на данных, введенных пользователем в текстовом поле формы, то добавьте значение переменной к остальной части строки. Если переменная представляет строковое значение, заключите ее в одинарные кавычки:

```
StrSQL = "SELECT Name FROM Kids WHERE Hates = ' "  &
frmInputForm.TextBoxl & " ' "
```

Если переменная представляет данные, а не строку, заключите ее между символами #.

Переменные, представляющие числовые значения, не требуют использования каких-либо открывающих и закрывающих символов.

```
Dim rs As Object
Dim rs As Recordset
Dim int As Integer ' nonobject variable declaration
Dim db As Database ' object variable declaration
InputBox(prompt [, title] [, default])
On Error Resume Next

Sub example()
        On Error GoTo ERR_EXAMPLE
        MsgBox rs.RecordCount
        Exit Sub
  ERR_EXAMPLE:
        MsgBox Err.Description, vbCritical
        Resume Next
End Sub
```

Методы запуска запросов

Макрокоманда, метод или объект	Объект запускается из	Источник	Пример
Макрокоманда runSQL	DoCmd	Строка SQL	DoCmd.runSQL strSQL
Макрокоманда OpenQuery	DoCmd	Заготовленный запрос	DoCmd.OpenQuery “qryTotals”
Метод Open Recordset (DAO)	Включение или БД	Строчка SQL или заготовленный запрос	db. OpenRecordset strSQL
Метод Open (DAO)	Включение или набор записей	Строка SQL, заготовленный запрос или процедура	rs.Open “SELECT * FROM t”
Метод Execute (DAO)	Включение или БД	Строка SQL	db.execute strSQL
Метод Execute (ADO)	Включение или команда	Строка SQL	cnn.execute strSQL
QueryDef (заготовленный запрос DAO)	Включение или БД	Заготовленный запрос	db.QueryDefs ("qryTotals")