Казанский государственный университет

Факультет вычислительной математики и кибернетики

Кафедра системного анализа и информационных технологий

Устюгова В.Н.

Практикум для изучения возможностей работы в СУБД Access

Учебно-методическое пособие

КАЗАНЬ 2010

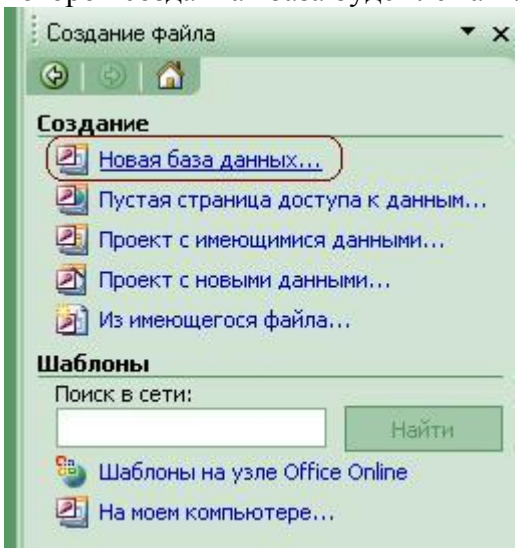
Представленные методические указания являются методическим пособием для проведения практических занятий и самостоятельного изучения системы управления базами данных (СУБД) MS ACCESS. В работе рассматриваются основные возможности MS ACCESS, предлагаются задания по практическому использованию этих возможностей. Методическое пособие может быть рекомендовано для студентов изучающих дисциплину «Базы данных».

Учебно-методическое пособие публикуется по решению учебно-методической комиссии факультета вычислительной математики и кибернетики КГУ от 13 мая 2010 года.

Работа в СУБД Access

Для выполнения лабораторных работ необходимо создать базу данных (например, **my_db.mdb**) в Microsoft Access (MS Access). Для этого запустите приложение MS Access и в меню «**Файл**» выберите пункт «**Создать**».

Далее указываем, что создавать будем новую базу данных (ссылка «**Новая база данных**». В открывшемся окне указываете имя файла базы данных **my_db.mdb** и папку, в которой созданная база будет лежать.



На рис.1 представлено окно MS Access для работы с базой данных **my_db.mdb.**

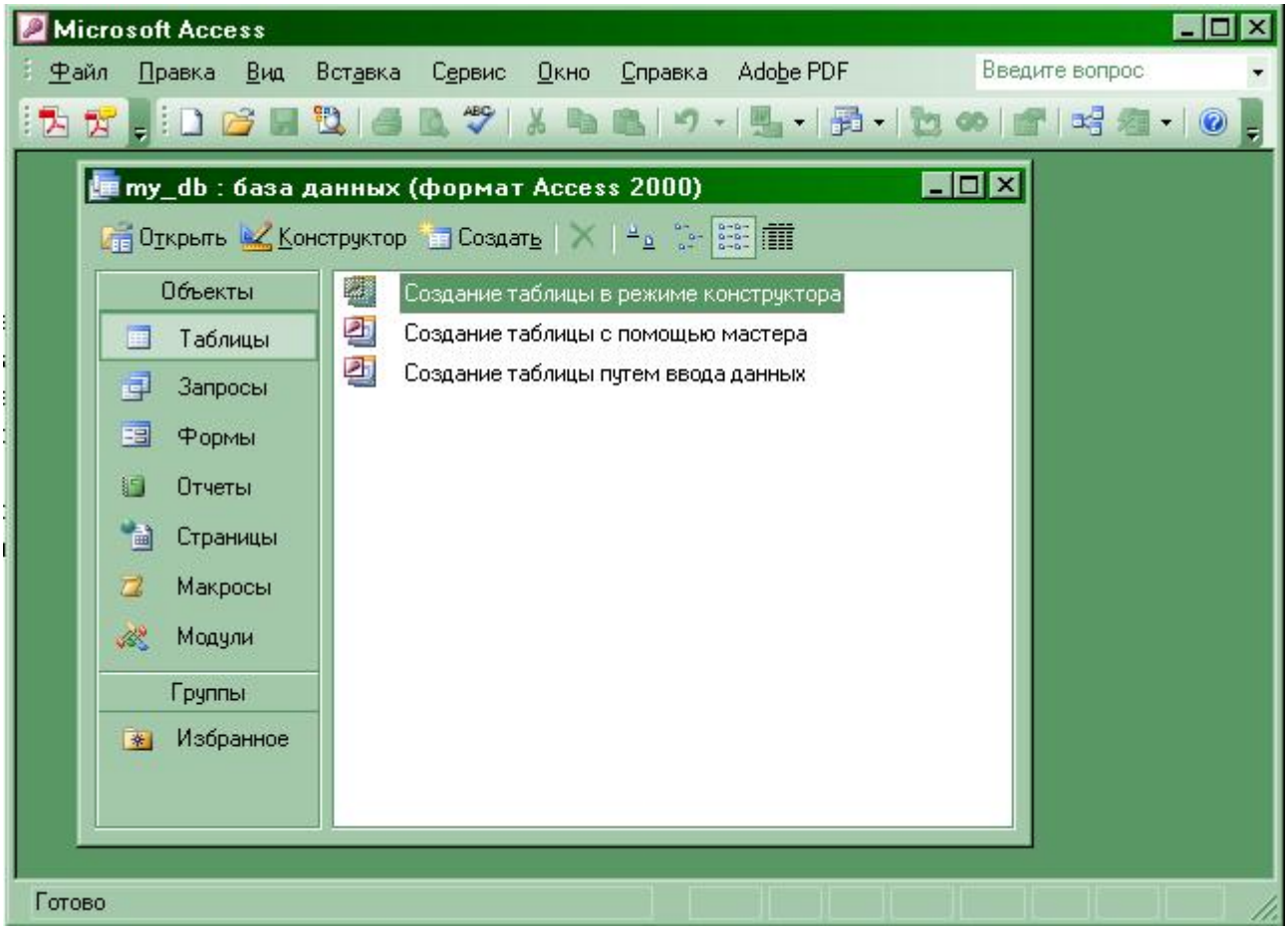


Рис.1. Окно MS Access для работы с базой данных **my_db.mdb**

Для того чтобы открыть существующую базу данных (БД), запустите приложение MS Access и в меню «**Файл**» выберите пункт «**Открыть**» и найдите файл базы данных **my_db.mdb на компьютере.**

В левой части окна базы данных на рис.1 перечислены «**Объекты**» MS Access. Объекты – это отдельные компоненты БД, которые используются для хранения и представления информации. Каждый объект имеет имя, которое может содержать до 64 символов, включая пробелы (не рекомендую использовать пробелы в именах). В MS Access основными объектами являются: таблицы, запросы, формы, отчеты, макросы и модули. Все объекты одной БД хранятся в общем файле с расширением mdb.

Чтобы открыть существующий объект используйте кнопку «**Открыть**». Для создания новых объектов следует использовать кнопку «**Создать**», а для модификации существующих объектов — кнопку «**Конструктор**».

Лабораторная работа №1

Работа с таблицами

Цель: Приобрести умения и навыки при работе с таблицами. Научиться создавать таблицы с помощью конструктора, задавать маску ввода для поля, научиться форматировать структуру и представление таблицы, научиться создавать ключи и индексы для таблицы, разобраться со связями между таблицами (схема данных), научиться импортировать, экспортировать и присоединять таблицы.

Таблица используется для хранения информации в БД.

Чтобы создать новую таблицу следует выбрать объект «**Таблицы**» и щелкнуть по кнопке «**Создать**» (рис.1)**.** MS Access открывает окно «**Новая таблица**» и предлагает

несколько способов создания таблицы (рис.2). Мы будем создавать и работать с таблицами в режиме конструктора.

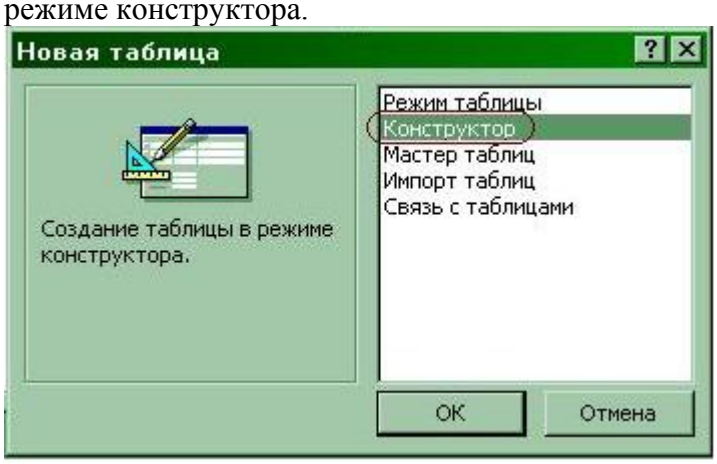


Рис.2. Создание новой таблицы

На рис.3 представлен еще один способ создания таблицы в режиме конструктора.

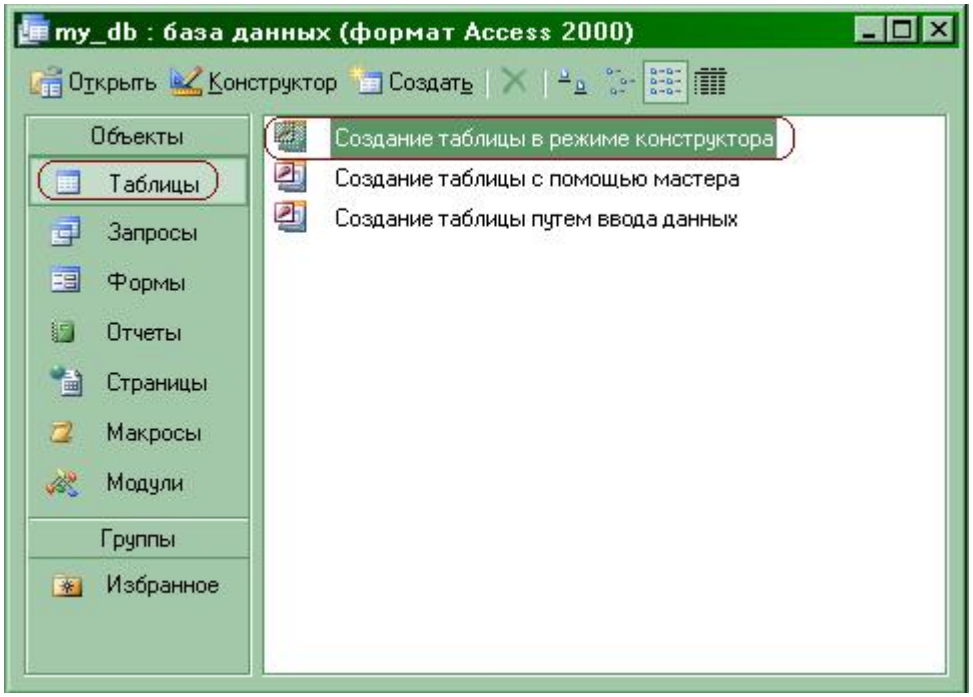


Рис.3. Создание таблицы в режиме конструктора

При создании таблицы в режиме конструктора открывается окно, представленное на рис.4.

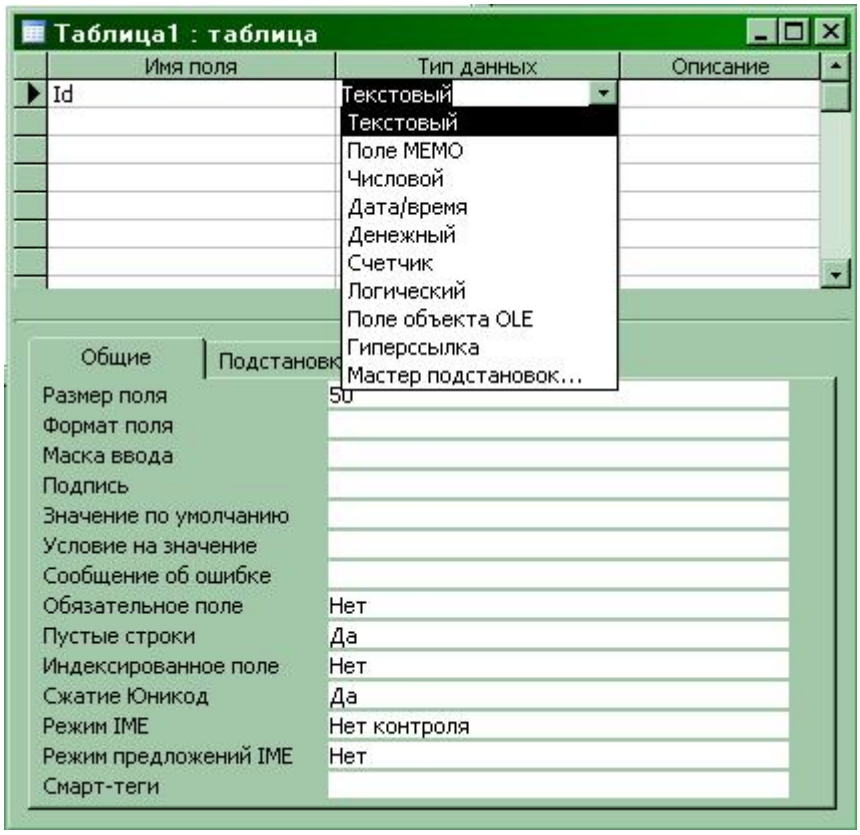


Рис.4. Работа с таблицей в режиме конструктора

В верхней части окна находится строки для полей таблицы, содержащей три графы: имя поля, тип данных и описание. Обязательно надо заполнять: «**Имя поля**» и «**Тип данных**». Для задания полей будущей таблицы следует выполнить следующие действия:

- определить имя поля таблицы (до 64 алфавитно-цифровых символов, включая пробелы);
- задать тип данных поля (щелкнуть левой кнопкой мыши в столбце «**Тип данных**» и выбрать тип из списка);
- ввести описание поля (необязательно);
- задать свойства поля (необязательно);

Основные типы данных:
текстовый — алфавитно-цифровые данные, до 255 байт;
поле MEMO — комментарии и пояснения, до 64 Кбайт;
числовой — числовые данные;
дата/время — значения даты и времени;
денежный — денежные значения;
счетчик — автоматически генерируются уникальные значения, пользователь не может менять значения в этом поле;
логический — логические значения типа Да/Нет, Истина/Ложь или Вкл/Выкл;
объект OLE — рисунок, таблица Excel, документ Word или другие данные в двоичном формате;
гиперссылка — строка, состоящая из букв и цифр и представляющая адрес гиперссылки.

Каждое поле имеет набор характеристик, называемых *свойствами*, которые задают условия хранения и отображения данных. Набор свойств зависит от типа данных поля. Для получения подробной справки по конкретному свойству нужно щелкнуть по интересующему свойству и нажать **F1.**

 1. Создать таблицу в режиме конструктора:

Сотрудники

Имя поля	**Тип данных**	**Свойства поля**
СотрудникId	счетчик	последовательные значения
Фамилия	текст	40 символов
Имя	текст	30 символов
Отчество	текст	30 символов
Пол	текст	1 символ
ДатаРождения	дата/время	краткий формат даты
Адрес	текст	50 символов
Телефон	текст	8 символов
Должность	текст	30 символов

Сохранить таблицу под именем Сотрудники. Первичный ключ задать, когда спросят. Варианты сохранения таблицы:

- меню «**Файл**» _ пункт «**Сохранить**» или
- меню «**Файл**» _ пункт «**Сохранить как**» или
- Закрыть окно с шаблоном таблицы.

 2. Заполнить таблицу 1 записью.

Чтобы открыть существующую таблицу, выбрать объект «**Таблицы**» и щелкнуть по кнопке «**Открыть**». Если Вы открыли таблицу в режиме конструктора, то переключиться в режим таблицы можно по меню «**Вид**» _ пункт «**Режим таблицы**» или выбрать кнопку режим таблицы на панели инструментов. Esc - отказ от ввода в поле. Esc, Esc - отказ от ввода новой записи.

 3. Задать маску ввода для поля телефон (в виде 12-34-56) и для поля ДатаРождения (в виде 01/12/1960). F1 (справка) на свойстве поля «**Маска ввода**». Опробовать заполнение полей по установленной маске ввода.

Свойство «**Маска ввода**» (InputMask)

Свойство Маска ввода задает маску ввода, облегчающую ввод данных в элемент управления - поле. Значение данного свойства определяется автоматически при использовании мастера по созданию масок ввода.

Значение свойства «**Маска ввода**» может содержать до трех разделов, разделяемых точкой с запятой (;).

Раздел1;Раздел2;Раздел3

Раздел1	Представляет саму маску ввода (например, !(999) 000-0000). Перечень символов, используемых для определения масок ввода приводится ниже в таблице.
Раздел2	Определяет режим занесения в таблицу строковых констант, добавляемых к символам, вводящимся пользователем. Введенный в данный компонент символ 0 указывает, что постоянные символы (например, скобки и дефисы в маске ввода телефонных номеров) сохраняются вместе с введенными пользователем символами; значение 1 или пустое значение данного компонента указывает, что сохраняются только символы, введенные пользователем.
Раздел3	Определяет символ, используемый для изображения пустых позиций в маске ввода, в которые помещаются вводящиеся пользователем символы. В этом компоненте можно указать любой символ ANSI; пробел необходимо заключить в кавычки ("").

В программах Visual Basic значение данного свойства задается при помощи строкового выражения. Например, следующая инструкция для поля определяет маску ввода для телефонных номеров:

Forms!Клиенты!Телефон.InputMask = "(###) ###-####"

При создании маски ввода пользователь имеет возможность указать, что часть данных следует вводить обязательно (например, региональный код для телефонных номеров), а другие данные являются необязательными (например, добавочный номер телефона). Эти символы определяют тип данных, например номер символа, который необходимо ввести для каждого символа маски ввода.

Символы, которые следует вводить в маску ввода, определяются следующими специальными символами.

Символ	Описание
0	Цифра (обязательный символ; знаки (+) и (-) не разрешены).
9	Цифра или пробел (необязательный символ; знаки (+) и (-) не разрешены).
#	Цифра или пробел (необязательный символ; незаполненные позиции выводятся как пробелы в режиме редактирования, но удаляются при сохранении данных; знаки (+) и (-) не разрешены).
L	Буква (обязательный символ).
?	Буква (необязательный символ).
A	Буква или цифра (обязательный символ).
a	Буква или цифра (необязательный символ).
&	Любой символ или пробел (обязательный символ).
C	Любой символ или пробел (необязательный символ).
. , : ; - /	Десятичный разделитель, разделители групп разрядов, времени или даты. (Используемые символы разделителей определяются настройками, выбранными в окне Язык и стандарты панели управления Windows).
<	Преобразует все символы к нижнему регистру.
>	Преобразует все символы к верхнему регистру.
!	Указывает, что маска ввода заполняется справа налево; этот символ следует использовать, если в левой части маски находятся позиции, заполнять которые не обязательно. Маски ввода обычно заполняются слева направо. Символ восклицательного знака можно помещать в произвольную позицию в маске ввода.
\	Указывает, что следующий символ следует воспринимать как постоянный (а не специальный) символ (например, \A представляет символ «A»). Примечание.

Для элемента управления значение данного свойства задается в окне свойств. Для поля в таблице или запросе значение данного свойства задается в режиме конструктора таблицы (в окне свойств поля) или в режиме конструктора окна запроса (в окне свойств поля). Кроме того, значение свойства Маска ввода (InputMask) можно задать в макросе или в программе Visual Basic.

В следующей таблице приводятся примеры часто используемых масок ввода и образцы значений, соответствующих этим маскам.

Маска ввода Образцы значений
(000) 000-0000 (206) 555-0248
(999) 999-9999 (206) 555-0248
( ) 555-0248
(000) AAA-AAAA (206) 555-TELE
#999 –20

2000
>L????L?000L0 GREENGR339M3
MAY R 452B7
>L0L 0L0 T2F 8M4
00000-9999 98115-
98115-3007
>L<????????????? Мария
Изаура
SSN 000-00-0000 SSN 555-55-5555
>LL00000-0000 DB51392-0493

4. Задать для поля Пол значение по умолчанию, например, М (свойство «**Значение по умолчанию**»).

5. Заполнить таблицу еще несколькими записями.

6. Вставить поле Паспорт (текст, 50 символов) перед полем Адрес.

В режиме конструктора встать на поле Адрес. Меню «**Вставка**» _ пункт «**Строки**». Ввести новое поле.

7. Перемещение полей. Поле Должность вставить перед полем Адрес.

В режиме конструктора отметить поле (Shift + поля для нескольких полей). Подвести мышь к перемещаемым полям и когда указатель мыши примет вид стрелки, удерживая нажатой левую кн. Мыши, переместить поле.

8. Удаление полей. Удалить поле Паспорт.

В режиме конструктора выбрать поле. Меню «**Правка**» _ пункт «**Удалить**» или клавиша Del.

9. Изменение макета таблицы.

Переключиться в режим таблицы. Меню «**Формат**» (пункты шрифт, ширина столбца, высота строки, расположить столбцы, скрыть столбцы, показать столбцы, закрепить столбцы, освободить все столбцы, ячейки (для вывода сетки)...)

10. Параметры, задаваемые Access по умолчанию можно изменить в меню «**Сервис**» _ пункт «**Параметры**» _ закладка «**Таблицы и Запросы**».

12. Удалить первую запить таблицы (выделить запись, Del или меню «**Правка**» _ пункт «**Удалить запись**»).

13. Создать резервную копию таблицы

В режиме конструктора меню «**Файл**» _ пункт «**Сохранить как**» ... задать новое имя таблицы (при этом сохранится шаблон таблицы).

Выделить все записи в старой таблице (меню «**Правка**» _ пункт «**Выделить все**»), сохранить их во внутреннем буфере (меню «**Правка**» _ пункт «**Копировать**»).

Для новой таблицы в режиме таблицы встать на первую запись и восстановить записи из внутреннего буфера (меню «**Правка**» _ пункт «**Вставить**»).

Внимание! Поля типа счетчик в новой таблице Access заполнит сам.

14. Переименовать новую таблицу в таблицу СотрудникиФирмыА.

Переименовать таблицу можно следующими способами:

- меню «**Правка**» _ пункт «**Переименовать**» или
- Правая кнопка мыши _ пункт «**Переименовать**» или

- Один раз щелкнуть на имени таблицы, переименовать.

 15. Удалить таблицу СотрудникиФирмыА:

В окне базы данных выделить таблицу. Клавиша Del или меню «**Правка**» _ пункт «**Удалить**».

Ключевые поля, индексы

Ключевое поле — это одно или несколько полей, комбинация значений которых однозначно определяет каждую запись в таблице. Если для таблицы определены ключевые поля, то MS Access предотвращает дублирование или ввод пустых значений в ключевое поле. Ключевые поля используются для быстрого поиска и связи данных из разных таблиц при помощи запросов.

Поле типа Счетчик предоставляет уникальные значения, поэтому поля этого типа могут быть сделаны ключевыми полями. Если до сохранения созданной таблицы ключевые поля не были определены и в таблице есть поле типа Счетчик, то при сохранении таблицы Access предложит поле типа Счетчик создать как ключевое поле.

Для создания простого ключа достаточно иметь поле, которое содержит уникальные значения (например, коды или номера). Если выбранное поле содержит повторяющиеся или пустые значения, его нельзя определить как ключевое.

Составной ключ необходим в случае, если невозможно гарантировать уникальность записи с помощью одного поля. Он представляет собой комбинацию нескольких полей.

 1. **Фамилия** и **Имя** сделать ключевыми полями.

Открыть таблицу в режиме конструктора и выделить эти поля при нажатой клавише Ctrl, далее меню «**Правка**» _ пункт «**Ключевое поле**» или кнопка «**Ключевое поле**»  на панели инструментов.

 2. Снять установку ключа с полей **Фамилия** и **Имя**, а поле СотрудникId сделать ключевым.

Чтобы удалить ключ, необходимо открыть таблицу в режиме конструктора, выбрать имеющееся ключевое поле (ключевые поля) и далее меню «**Правка**» _ пункт «**Ключевое поле**» или кнопка «**Ключевое поле**»  на панели инструментов, при из области выделения должен исчезнуть значок (значки) ключевого поля.

С целью ускорения поиска и сортировки данных в любой СУБД используются индексы. **Индекс** является средством, которое обеспечивает быстрый доступ к данным в таблице на основе значений одного или нескольких полей (столбцов). Индекс представляет собой внутреннюю таблицу, в которой находится упорядоченный список значений и ссылок на те записи, в которых хранятся эти значения. Чтобы найти нужные записи, СУБД сначала ищет требуемое значение в индексе, а затем по ссылкам быстро отбирает соответствующие записи. Индексы бывают двух типов: простые и составные. **Простые** индексы представляют собой индексы, созданные по одному полю. Индекс, построенный по нескольким полям, называется **составным**. Примером составного индекса может быть индекс, построенный по столбцам «Фамилия» и «Имя».

Однако применение индексов приносит не только преимущества, но и недостатки. При добавлении и удалении записей или при обновлении значений в индексном столбце требуется обновлять индекс, что при большом количестве индексов в таблице может замедлять работу. Поэтому индексы обычно рекомендуется создавать только для тех столбцов таблицы, по которым наиболее часто выполняется поиск записей. В MS Access индексы хранятся в том же файле базы данных, что и таблицы и другие объекты MS Access.

Индексировать можно любые поля, кроме МЕМО-полей, полей типа Гиперссылка и объектов OLE.

Ключевые поля таблицы индексируются автоматически.

Для создания индекса по одному полю можно воспользоваться свойством поля «**Индексированное поле**» в таблице. При индексировании поля имеется две возможности. Выбор варианта «**Да (Совпадения не допускаются)**» означает, что создается уникальный индекс. В этом случае таблица не может иметь в этом поле повторяющиеся значения. При выборе варианта «**Да (Совпадения допускаются**)» создается индекс, учитывающий возможность повторения значений в этом поле.

 3. Создать индекс **Инд1** по полю Фамилия.

Открыть таблицу в режиме конструктора. Меню «**Вид**» _ пункт «**Индексы**». В столбец «**Индекс**» ввести имя индекса, в столбце «**Имя поля**» выбрать название поля из списка, в столбце «**Порядок сортировки**» указать как будут отсортированы значения поля в таблице индекса (по возрастанию или по убыванию).

 4. Создать индекс **Инд2** по полю Имя.

 5. Создать индекс **Инд** по двум полям **Фамилия** и **Имя.**

В столбец «**Индекс**» ввести имя индекса, в первой строке столбца «**Имя поля**» выбрать из списка название первого поля, в столбце «**Порядок сортировки**» указать как будут отсортированы значения поля в таблице индекса. В следующей строке столбца «**Имя поля**» выбрать из списка название второго поля, в столбце «**Порядок сортировки**» указать как будут отсортированы значения поля в таблице.

 6. Определить чем отличаются индексы Инд, Инд1, Инд2 (обратите внимание на «**Свойства индекса**») для каждого из индексов.

 7.Удалить индекс Инд1.

(В окне «**Индексы**» (таблице индексов для конкретной таблицы) выделить индекс и нажать клавишу «**Del**»).

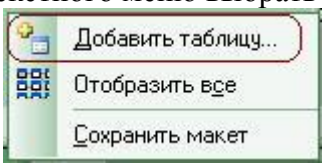
Объединение таблиц в схему данных

Для того чтобы было удобно просматривать, создавать, удалять и модифицировать связи между таблицами, в Microsoft Access используется схема данных.

Рис.5. Пример схемы данных

Чтобы открыть схему данных, необходимо выполнить команду меню «**Сервис**» _ пункт «**Схема данных**» (Tools, Relationships). По умолчанию схема данных будет содержать все таблицы со связями.

Чтобы добавить в схему данных таблицу, у которой связи еще не установлены, следует щелкнуть правой кнопкой мыши на свободном пространстве схемы данных и из контекстного меню выбрать команду «**Добавить таблицу**» (Show table).



В диалоговом окне «**Добавление таблицы**» (Show table) раскрыть вкладку «**Таблицы**» (Tables), выбрать из списка таблицу и нажать кнопку «**Добавить**» (Add) (рис.6). Затем закрыть диалоговое окно по кнопке «**Закрыть**».

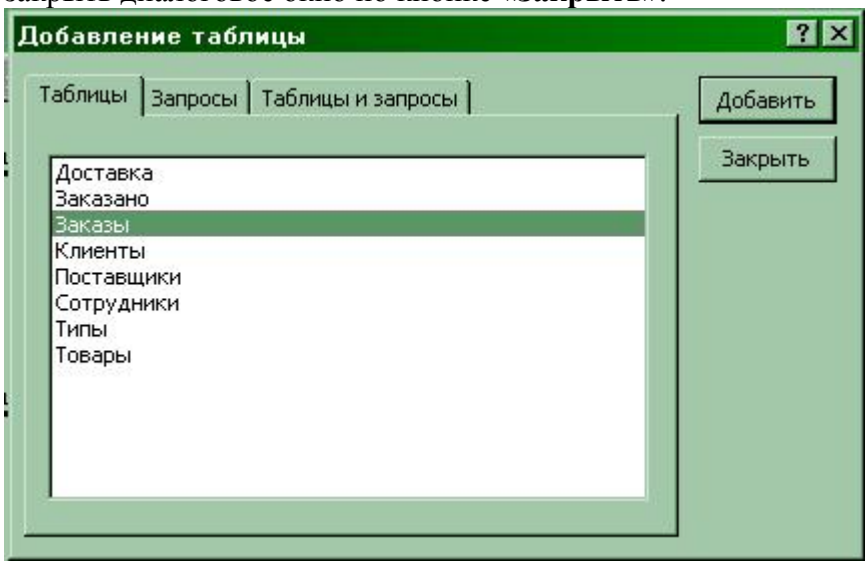


Рис.6. Добавление таблицы в схему данных

Чтобы убрать какую-либо таблицу из схемы данных, необходимо щелкнуть левой кнопкой мыши на любом месте этой таблицы и нажать клавишу «**Delete**» (удаление таблицы из схемы данных не означает удаление ее из базы данных).

Связь между таблицами MS Access строит автоматически, если две таблицы имеют одинаковые названия связанных полей и согласованные типы данных, причем хотя бы в одной из таблиц связанное поле является ключевым.

Если нужная связь автоматически не создана, следует выбрать в главной таблице поле для связи, нажать левую кнопку мыши и перетащить поле во вторую таблицу. Отпустить левую кнопку мыши над тем полем подчиненной таблицы, с которым устанавливается связь. После этого появится диалоговое окно «**Изменение связей**» (Edit Relationships)

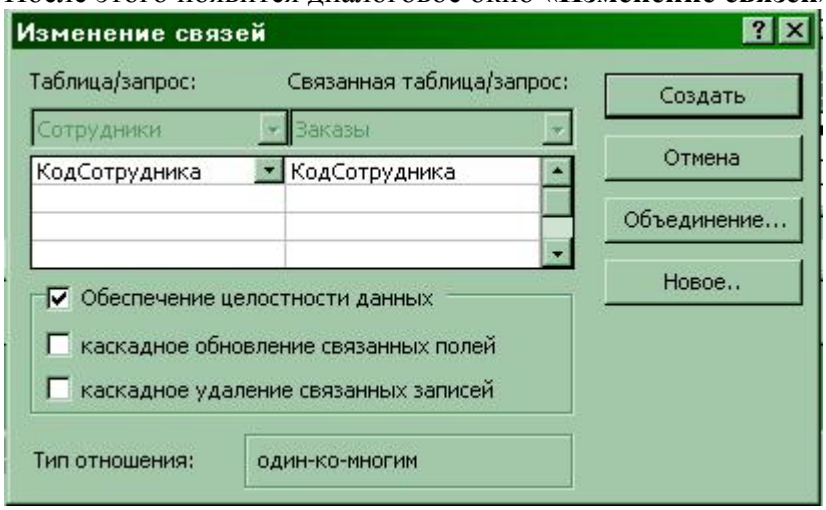


Рис.7. Изменение связей между таблицами

Когда открыто окно «**Схема данных**» (Relationships) можно изменить связь из меню «**Связи**» _ пункт «**Измененить связь**» (рис. 8).

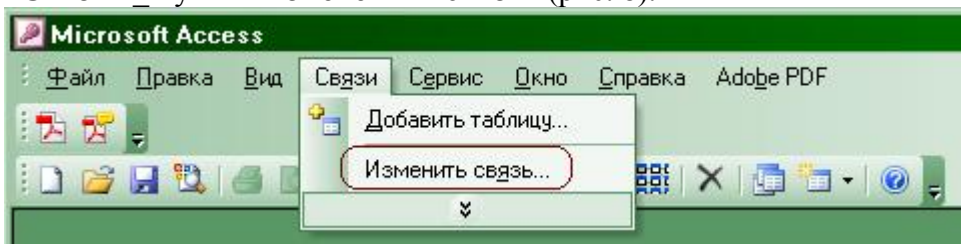


Рис.8. Изменение связей между таблицами

Режим обеспечения целостности данных для конкретной связи можно включить, если выполнены следующие условия:

- поле связи главной таблицы является первичным ключом или имеет уникальный индекс;
- связанные поля имеют один и тот же тип данных;
- обе связанные таблицы принадлежат одной базе данных Access.

Если включен режим «**Обеспечения целостности данных**», то можно дополнительно указать, следует ли автоматически выполнять для связанных записей операции каскадного обновления и каскадного удаления. Если включить режим «**Каскадное обновление связанных полей**», то при изменении значения ключа в главной таблице будут автоматически обновлены соответствующие значения в связанных записях подчиненной таблицы. Если включить режим «**Каскадное удаление связанных записей**» при удалении записи из главной таблицы будут автоматически удалены связанные с ней записи в подчиненной таблице.

В том случае, когда эти режимы не включены, а режим обеспечения целостности данных включен, MS Access не позволит изменить значение в ключевом поле главной таблицы, а также удалить запись в главной таблице, если в подчиненной таблице имеются данные, связанные с этой записью.

Если включен режим обеспечения целостности данных, то MS Access изобразит на конце линии, соответствующей главной таблице, цифру 1. На другом конце линии,

соответствующем подчиненной таблице, будет изображен символ бесконечности ∞ для связи типа «один-ко-многим» и цифра 1 для связи типа «один-к-одному».

Чтобы изменить существующую связь между таблицами следует щелкнуть левой кнопкой мыши на линии связи и нажать на правую кнопку мыши (рис. 9).

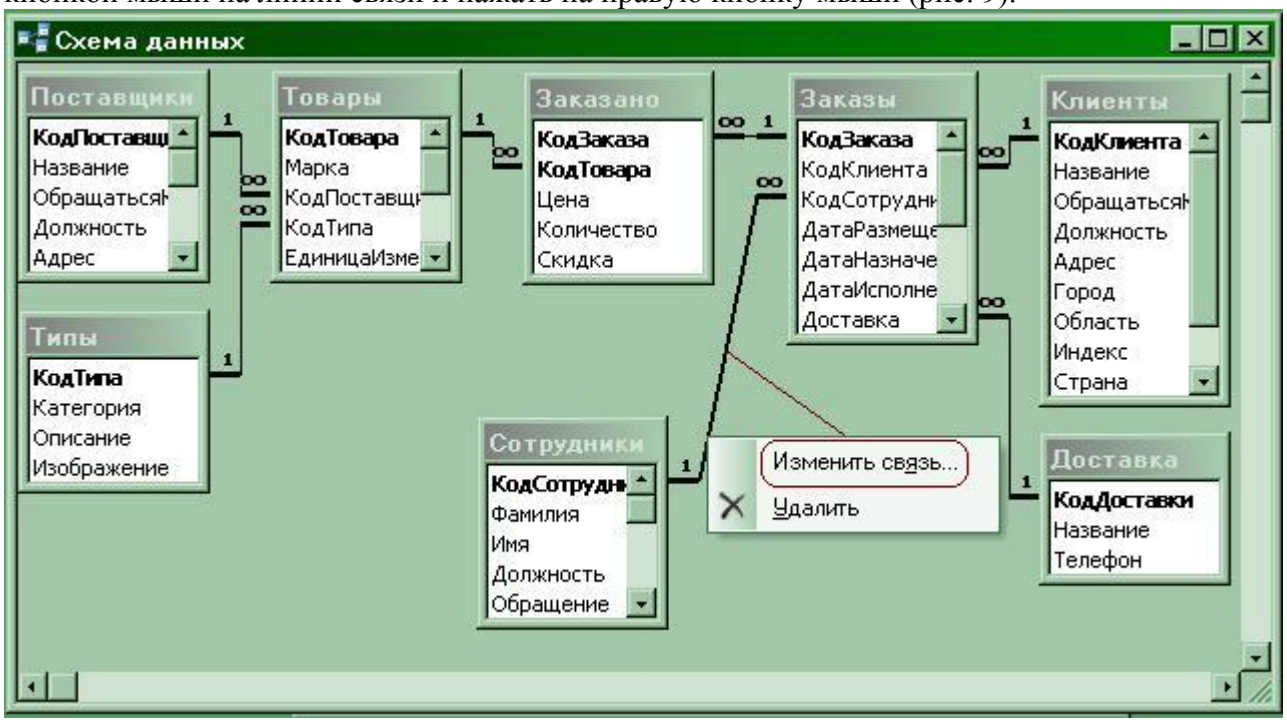


Рис.9. Изменение существующей связи между таблицами

Чтобы изменить параметры объединения таблиц следует в диалоговом окне «**Изменение связей**» нажать на кнопку «**Объединение…**», выбрать вариант объединения и нажать на кнопку «**ОК**».

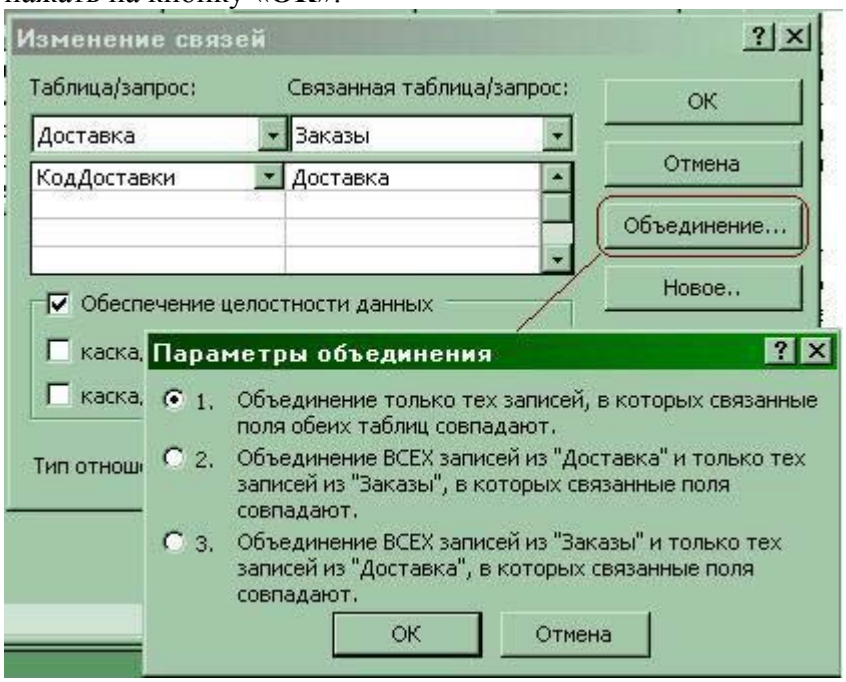


Рис.10. Изменить параметры объединения

1. Создать с помощью конструктора таблицу **Дети**:

Имя поля	**Тип данных**	**Свойства поля**
Id	счетчик	

СотрудникId	числовой	
Имя	текст	30 символов
Фамилия	текст	30 символов
ДатаРождения	дата/время	средний формат даты

«**ОК**».

 2. Заполнить таблицу тремя записями.

В одной из записей поле **СотрудникId** сделать на 1 больше, чем максимальное значения поля **СотрудникId** в таблице Сотрудники.

 3. Меню «**Сервис**» _ пункт «**Схема данных**». Добавить в схему данных таблицы Сотрудники и Дети.

 4. Связать эти таблицы по полю **СотрудникId** (одно поле на другое перенести мышью). Выбрать обеспечение целостности данных, кнопка «**Создать**». Что происходит? Если вы правильно выполнили все в пункте 2 (т.е. в поле **СотрудникId** таблицы Дети внесли значение отличное от любого значения поля **СотрудникId** таблицы Сотрудники), то появится сообщение, что ваши данные нарушают целостность данных. Измените значение поля **СотрудникId** таблицы Дети на допустимое и повторите пункт 4.

 5. Удалить из таблицы сотрудники запись, на которую ссылается запись из таблицы Дети. Что происходит? (MS Access поддерживает целостность данных).

 6. В режиме конструктора в таблице Сотрудники измените тип поля **СотрудникId** на числовой. Что происходит? (MS Access следит за установленными связями). Тип любого другого поля, которое не задействовано в связи изменить можно (если в это поле не вводили значения предыдущего типа или предыдущий тип MS Access может преобразовать в текущий тип). Удалите связь в схеме данных, а потом в таблице Сотрудники измените тип поля **СотрудникId** на числовой. Поле оставьте ключевым. Еще раз выполните пункт 4.

 7. Измените значение поля **СотрудникId** таблицы Сотрудники в записи, на которую ссылается запись из таблицы Дети. Что происходит? (MS Access поддерживает целостность данных).

 8. В схеме данных выделить связь (1 раз нажать на левую кнопку мыши). Нажать на правую кнопку мыши, выбрать **Изменить связь**, добавить каскадное удаление. В таблице Сотрудники удалить запись, на которую ссылается запись из таблицы Дети. Что происходит? (В таблице Дети должны удалиться записи, ссылающиеся на запись в таблице Сотрудники).

 9. В схеме данных выделить связь (1 раз нажать на лев. Кнопку мыши). Нажать на правую кнопку мыши, выбрать «**Изменить связь**», выбрать каскадное обновление, кнопка «**Создать**». В таблице Сотрудники изменить значение поля **СотрудникId** в записи, на которую ссылается запись из таблицы Дети. Что происходит в таблице Дети? (В таблице Дети значение поля **СотрудникId** в записях, ссылающихся на изменяемую запись должно измениться).

 10. Удалить связь между таблицами в схеме данных (выделить связь, клавиша «**Delete**»).

 11. Удалить таблицу из схемы данных (выделить таблицу, клавиша «**Delete**»).

Импорт таблиц

Таблицу можно создать, импортируя данные из файлов других форматов.

1. Из базы данных Борей (Борей.mdb или nwind.mdb) импортировать таблицу Сотрудники. Меню «**Файл**» _ пункт «**Внешние данные**» _ «**Импорт**». Далее выбрать папку, тип файла, имя файла, кнопка «**Импорт**». В окне «**Импорт объектов**» на закладке «**Таблицы**» выделить импортируемые таблицы и нажать кнопку «**ОК**» (рис. 11-12).

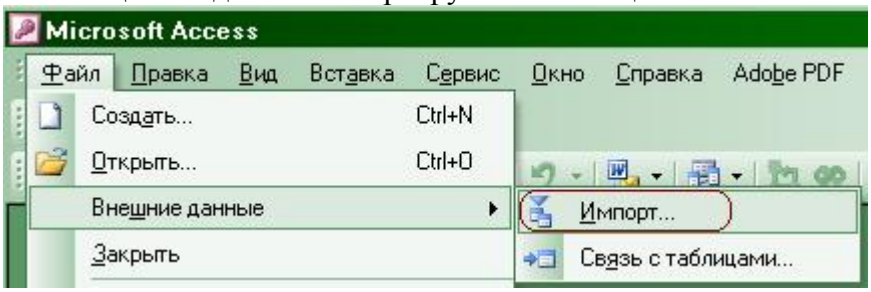


Рис.11. Импорт данных

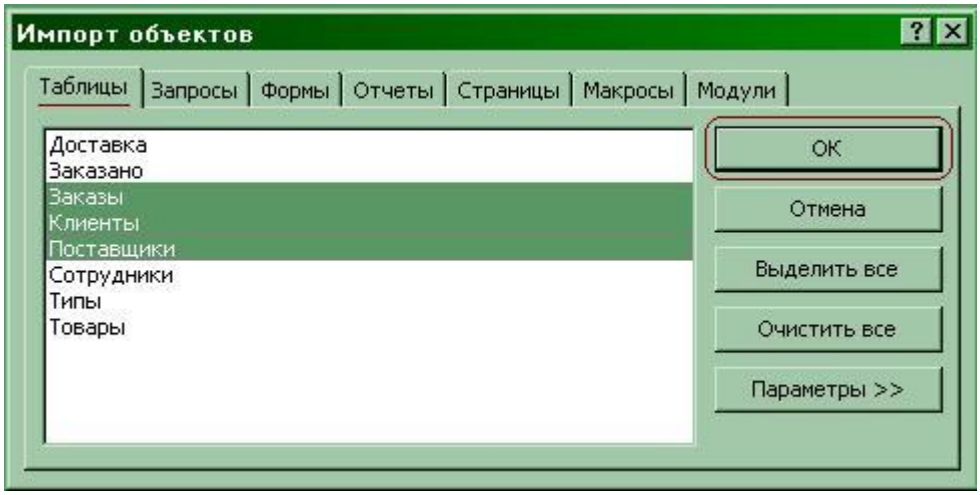


Рис.12. Выбор таблиц для импорта

Меню «**Сервис**» _ пункт «**Схема данных**»

Экспорт таблиц

MS Access позволяет экспортировать таблицу Access в другую базу данных (типа Access, dBase, FoxPro, Paradox) или файл другого формата (текстовый, документ Word, таблица Excel, документ HTML).

1. Экспортировать таблицу **Сотрудники1** во внешний файл данных формата Paradox. Выбрать таблицу в базе данных, меню «**Файл**» _ пункт «**Экспорт**», далее выбрать тип файла для сохранения, кнопка «**Экспорт**» (рис. 13-14).

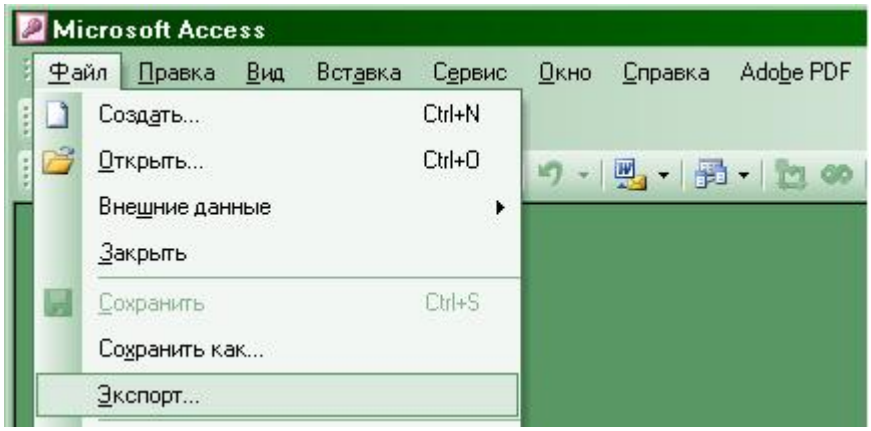


Рис.13. Экспорт таблицы

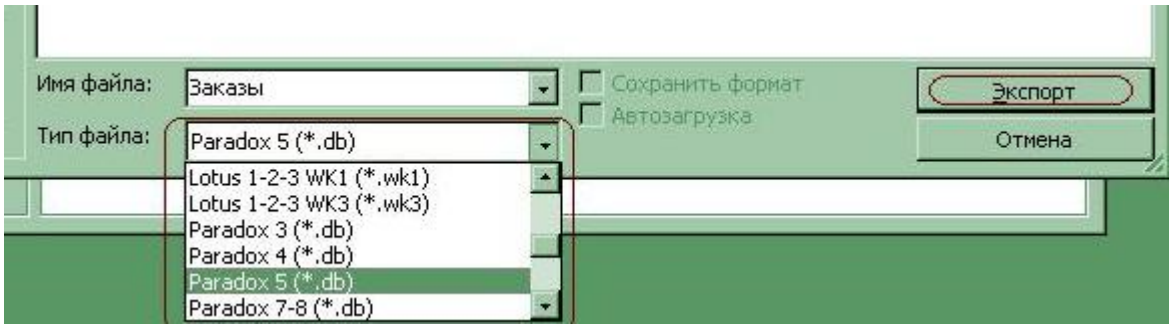


Рис.14. Продолжение экспорта таблицы

Присоединение таблиц

При импорте данных создается их копия в новой таблице текущей БД. После этого всякая связь с внешними данными теряется. Связывание (установление связи с внешними данными) позволяет использовать данные из внешнего файла (баз данных, электронных таблиц или текстовых файлов), не импортируя их в MS Access, и в этом случае пользователь всегда имеет дело с самой «свежей» информацией. Формат данных файла-источника не меняется и его можно продолжать использовать в приложении, в котором он был создан.

В тех случаях, когда внешний файл часто изменяется или используется в режиме коллективного доступа, целесообразно вместо операции импорта использовать связывание. Однако это может привести к определенному снижению быстродействия при работе с данными, так как MS Access максимально эффективно работает со своей копией данных и в «родном» формате.

1. Присоединить таблицу Сотрудники1 формата Paradox.
Меню «**Файл**» _ пункт «**Внешние данные**» _ «**Связь с таблицами**», выбрать файл соответствующего формата, нажать на кнопку «**Связь**».

Лабораторная работа №2

Работа с запросами

Цель: Приобрести умения и навыки при работе с запросами. Научиться создавать запросы с помощью конструктора, познакомиться с различными видами запросов.

1. Импортировать все таблицы из базы **Борей.mdb** или **nwind.mdb** в свою базу данных.

С помощью запроса можно:

- отобрать поля (взять часть полей из одной или нескольких таблиц),
- отобрать записи (включить условия),
- отсортировать данные,
- задать вопрос о данных, хранящихся в разных таблицах (Access, FoxPro, Paradox, dBase, Btrieve, SQL),
- выполнить вычисления,
- использовать запрос в качестве источника данных для форм, отчетов и других запросов,
- изменять табличные данные (обновлять, удалять, добавлять, создавать новые таблицы).

MS Access позволяет выполнять следующие типы запросов:

- запрос на выборку;
- запрос на создание таблицы;
- запрос на обновление;
- запрос на добавление записей;
- запрос на удаление записей;
- перекрестный запрос.

Чтобы создать новый запрос следует выбрать объект «**Запросы**» и щелкнуть по кнопке «**Создать**» (рис.15)**.** MS Access открывает окно «**Новый запрос**» и предлагает несколько способов создания запроса (рис.16). Мы будем создавать и работать с запросами в режиме конструктора.

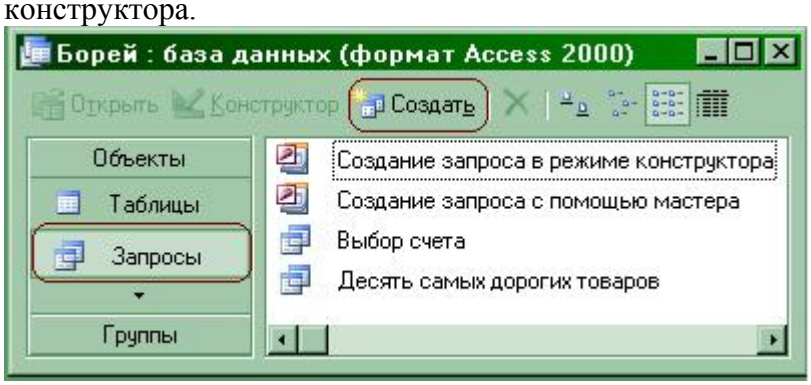


Рис.15. Создание запросов

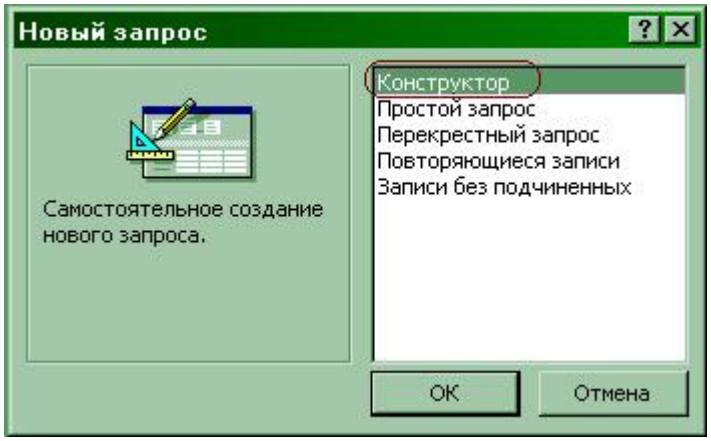


Рис.16. Разные способы создания запроса

Создание запросов на выборку

Далее на экране появится пустой бланк запроса с типом «**Запрос на выборку**» и окно «**Добавление таблицы**» для выбора таблиц или запросов, на основе которых будет построен

новый запрос (рис. 17). Следует выбирать имена таблиц или запросов и нажимать на кнопку «**Добавить**». Закрыть окно добавления таблиц по кнопке «**Закрыть**».

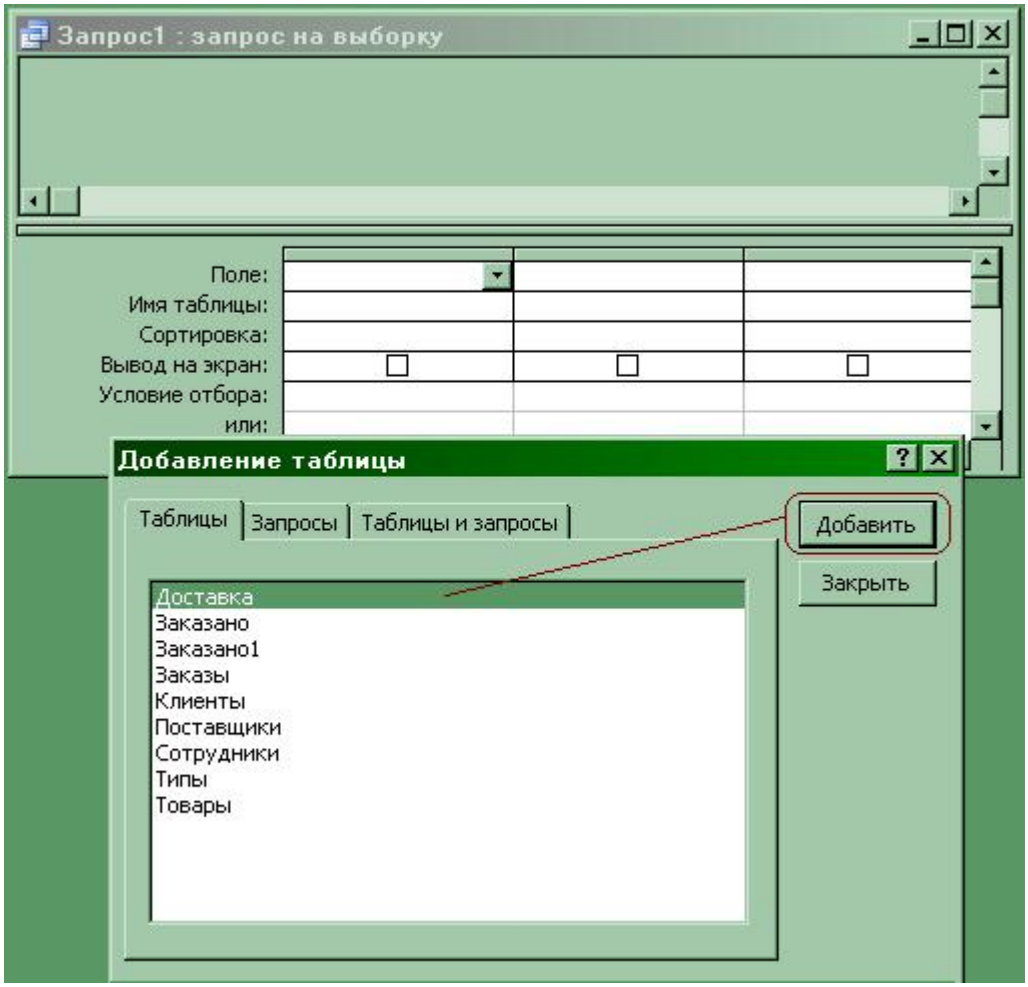


Рис.17. Добавление таблиц в бланк запроса

После того как таблицы будут добавлены, они появятся в бланке запроса (рис.18).

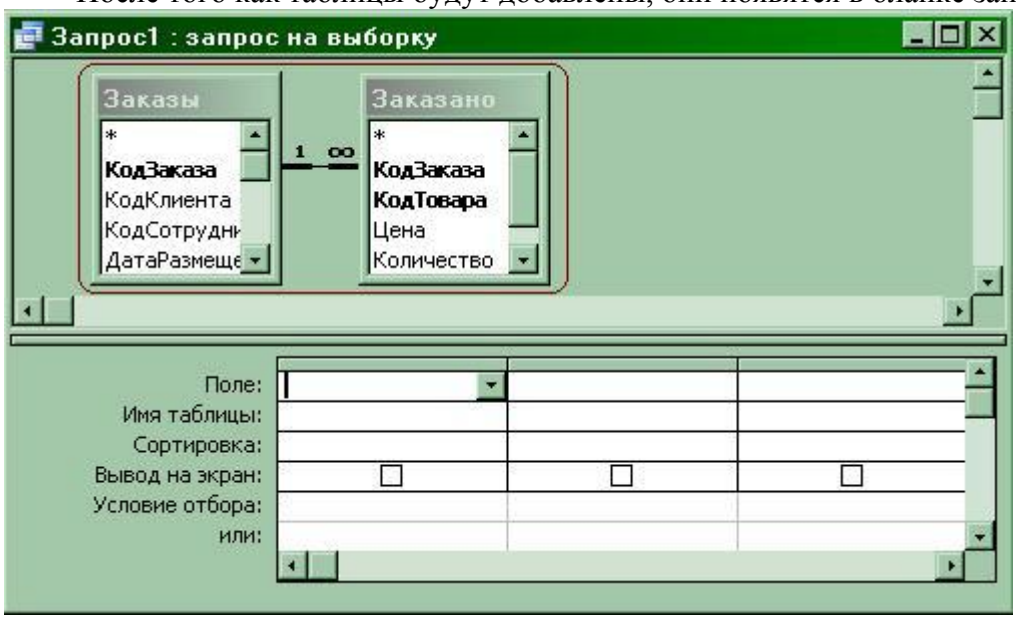


Рис.18. Добавили таблицы в бланк запроса

Далее в заданиях будет рассказано, как создавать запросы соответствующих типов.

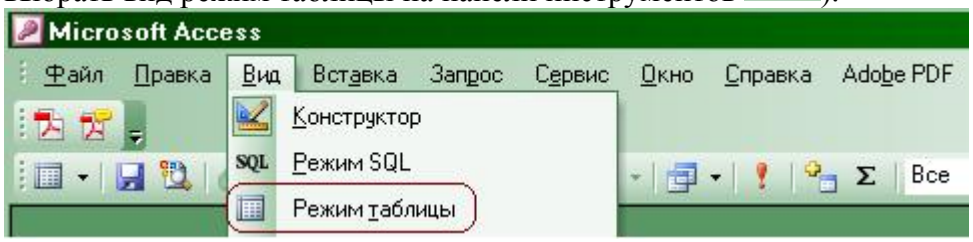
2. Создать Запрос для отражения информации о том «Какие поставщики какие товары поставляют».

Объект «**Запросы**», кнопка «**Создать**», выбрать Конструктор. Добавить таблицы **Поставщики** и **Товары** по кнопке «**Добавить**». Закрыть окно добавления таблиц. Установить связь между таблицами по полю **КодПоставщика** (перетащить мышью поле из одной таблицы и бросить его на соответствующее поле другой таблицы).

Если таблицы завязаны в схему данных, то линия объединения появится автоматически. MS Access сам может объединять таблицы автоматически, если есть поля с одинаковыми именами и типами. **!!!Внимание.** Проверяйте связи, если поля разного смысла названы одинаково.

Выбрать поля **Название** (табл. Поставщики), **Марка** и **КодТипа** (табл. Товары). 2 раза щелкнуть мышью на имени поля или перенести мышью имя поля в нижнюю часть бланка запроса.

Просмотреть запрос в режиме таблицы (меню «**Вид**» _пункт «**Режим таблицы**» или выбрать вид режим таблицы на панели инструментов).



Сохранить запрос (просто закрыть окно с запросом). Дать название запросу **ПоставщикиТоваров**.

3. Изменить ширину поля **Марка** (мышью сдвинув границу поля названия или через пункт меню «**Формат**», который появляется, если вы переключили запрос в режим таблицы) так, чтобы часть полей стали не видны в запросе.

4. Закрепить столбец **Название** (выделить столбец - 1 раз щелкнуть мышью на имени столбца) и из меню «**Формат**» выбрать пункт «**Закрепить столбцы**». Переместиться на поле **КодТипа** в любой записи. Освободить столбец **Название**.

5. Изменить название поставщика, который поставляет несколько товаров. Сохранить изменяемую запись (перейти к следующей записи).Что происходит? Открыть таблицу Поставщики и найти в ней поставщика название которого изменили. Обратите внимание на то, что закрепленный столбец не перемещается.

6. В запросе **ПоставщикиТоваров** изменить марку товара того же поставщика. Что происходит? Открыть таблицу **Товары** и найти в ней измененную марку товара.

7. В запросе **ПоставщикиТоваров** переименовать поле **Название** в **ИмяПоставщика**. В строке «**Поле**» в колонке **Название** ввести:

ИмяПоставщика**:** Название

Просмотреть запрос в режиме таблицы. Что произошло с именем поля?

8. В запросе **ПоставщикиТоваров** отсортировать поле **ИмяПоставщика** по возрастанию. В строке «**Сортировка**» выбрать «**по возрастанию**». Просмотреть запрос в режиме таблицы. Поле **КодТипа** отсортировать «**по убыванию**». Просмотреть запрос в режиме таблицы. Обратите внимание на то, что в этом поле **КодТипа** у одного и того же поставщика как бы нет сортировки «**по убыванию**».

9. Открыть таблицу **Товары** в режиме конструктора. Найти поле **КодТипа**. Какого оно типа? (числового). Обратите внимание, что данное поле в свойстве поля «**Подпись**» имеет значение Тип. Переключитесь в режим таблицы. Найдите поле **КодТипа**. Нашли? Это поле в режиме таблицы имеет подпись **Тип**. Какого типа поле Тип по данным? (строкового) Почему? Переключитесь в режим конструктора, выберите поле и в свойствах поля переключитесь на закладку «**Подстановка**». Реально это поле числовое и сортировка произошла по числам, а не по подстановочным значениям.

10.Создать запрос, позволяющий «Найти клиентов, которые ничего не заказали».
В запросе связать таблицы **Клиенты** и **Заказы**. Выбрать поля **Название** (из Клиенты) и **КодСотрудника**, **ДатаРазмещения** (из Заказы). Использовать параметры объединения (всех из Клиенты и только тех и из Заказы, в которых связанные поля совпадают). Выбрать связь (линию, связывающую таблицы), 1 раз щелкнуть левой кнопкой мыши, 1 раз щелкнуть правой кнопкой мыши, выбрать «**Параметры объединения**», определить тип объединения. В строке «**Условие отбора**» для поля **КодСотрудника** ввести null. Просмотреть запрос в режиме таблицы. Сохранить запрос под именем **КлиентыБезЗаказов**.

Для поиска полей, которые **содержат значения**, следует использовать **not null** или **is not null** или **not is null**.

Для поиска полей, которые **не содержат значений**, следует использовать **null** или **is null**.

11. Аналогичный запрос можно создать и другим способом. Это задание можно будет выполнить, если на вашем компьютере установлены дополнительные возможности Мастера MS Access!!!

Выполнить: Объект «**Запросы**», кнопка «**Создать**», выбрать «**Записи без подчиненных**». Выбрать таблицу **Клиенты**, кнопка «**Далее**», выбрать таблицу **Заказы,** кнопка «**Далее**», связать таблицы по полю **КодКлиента**, кнопка «**Далее**», выбрать поля **Название** и **ОбращатьсяК**, кнопка «**Далее**», следовать за мастером запросов. Просмотреть запрос в режиме таблицы и конструктора. Сохранить запрос.

Для задания диапазона значений в окне конструктора запросов могут быть использованы операторы **>** (больше), **>=** (не менее), **<** (меньше), **<=** (не более) и **Between** (Выражение1) **and** (Выражение2) как с текстовыми и числовыми полями, так и с полями дат). Для ввода условия выборки можно использовать окно «**Построитель выражений**» ().

12. Создать запрос, который позволяет «Найти клиентов, разместивших заказ после 1 января 1995г. »
Создать копию запроса **КлиентыБезЗаказов** (меню «**Файл**» _ пункт «**Сохранить как**») и откорректировать новый запрос. В строке «**Условие отбора**» для поля **ДатаРазмещения** ввести >01/01/1995.

13. Создать запрос, который позволяет «Найти клиентов, разместивших заказы после 1 января 1995г, чей заказ исполнен меньше, чем за 10 дней».

Создать копию запроса **КлиентыБезЗаказов** и откорректировать новый запрос. Добавить поле **Дата исполнения**. В строке «**Условие отбора**» для поля **ДатаИсполнения** ввести <#01/01/1995#+10.

14. Создать запрос, который позволяет «Найти клиентов, разместивших заказ между 1 января 1995г и 31 января 1995г».

Использовать оператор **Between** (Выражение1) **and** (Выражение2) (смотри help).

15. Создать запрос, который позволяет «Вычислить на какую сумму было заказано каждого товара по каждому заказу».

В запросе связать таблицы **Заказано** и **Заказы**. Выбрать поля **Цена**, **Количество**, **КодЗаказа**. В свободную колонку строки «**Поле**» ввести Цена*Количество. Просмотреть запрос в режиме таблицы.

16. Выяснить какое числовое значение (не подстановочное) содержится в поле **КодТовара** предыдущего запроса.

Создать копию предыдущего запроса. В этот запрос добавить поле **КодТовара** и создать новое поле, в которое внести КодТовара+0. Просмотреть запрос в режиме таблицы. Ввиду того, что тип поля **КодТовара** числовой, то мы можем выполнять арифметические операции со значениями в этом поле. Строковые значения в поле **КодТовара** подставляются из другой таблицы из-за подстановки в этом поле (таблица **Заказано** в режиме конструктора, свойства поля **КодТовара,** значение из «**Источник строк**»**).**

Групповые операции

Запросы позволяют не только выбирать записи из таблиц, но и выполнять различные статистические функции. MS Access использует для этих целей «**Групповые операции**». Для того, чтобы в запросе организовать группировку и использовать групповые операции, следует выполнить Меню «**Вид**» _ пункт «**Групповые операции**» (рис. 19).

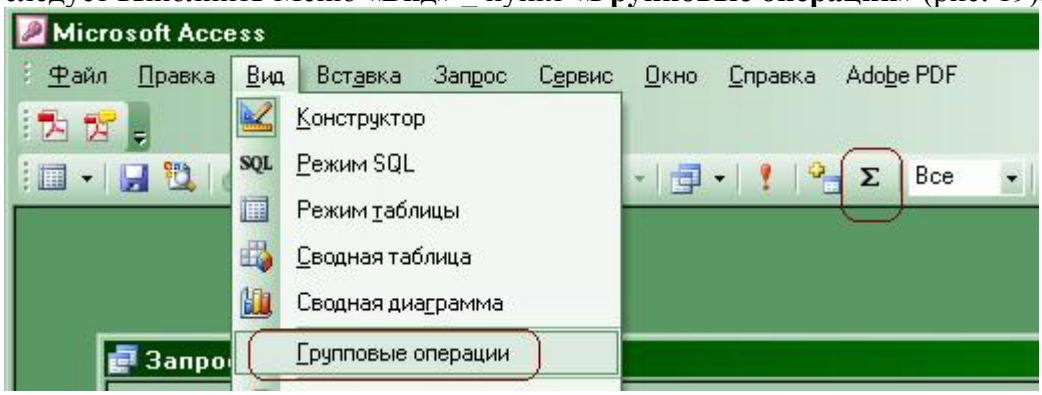


Рис.19. Добавление групповых операций в запрос

Если добавили «**Групповые операции**» к запросу, то в бланке запроса появится дополнительная строка «**Групповая операция**» (рис. 20). Для каждого поля из запроса следует из списка выбрать соответствующую групповую операцию (таблица 1).

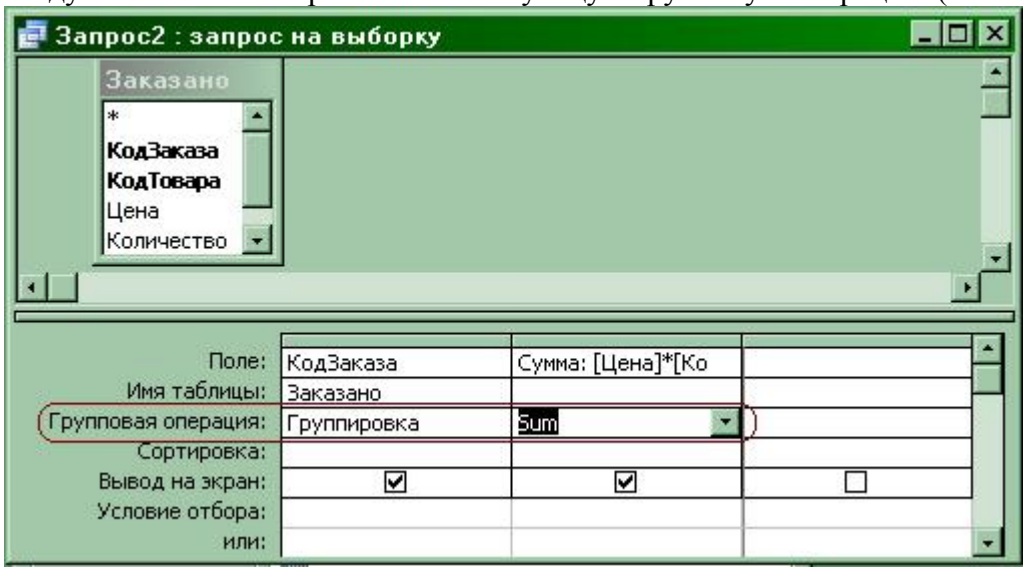


Рис.20. Использование групповых операций в запросе

Таблица 1. Доступные групповые операции

Название	**Функция**
Условие(Where)	Режим задания условия отбора для поля, по которому не выполняется группировка. Access автоматически делает такое поле скрытым
Выражение (Expression)	Вычисляемое поле, значение которого рассчитывается по сложной формуле
Группировка (Group By)	Поле, определяющее группу записей, по которой вычисляются статистические параметры. К одной группе относятся все записи, для которых значения поля с режимом Group By (Группировка) одинаковы
Last	Последнее значение в группе
First	Первое значение в группе
Var	Вариация значений поля
StDev	Стандартное отклонение величин ноля от среднего
Count	Количество записей, соответствующее ноле которых не содержит величины Null
Max	Максимальное значение
Min	Минимальное значение
Avg	Среднее значение поля
Sum	Сумма значений поля по всем записям

 17. Создать запрос, который позволяет «Вычислить на какую сумму сделан каждый заказ».

Создать запрос на базе таблицы **Заказано**, выбрать поле **КодЗаказа**, вычислить сумму по каждой позиции заказа (Цена* Количество) в созданном поле **Сумма**. Меню «**Вид**» _ пункт «**Групповые операции**». В строке «**Групповая операция**» в запросе для поля **Сумма** выбрать операцию **Sum**. Просмотреть запрос в режиме таблицы.

 18. Создать запрос, который позволяет узнать «Сколько клиентов какое количество заказов сделало с 1 января 1995г».

В запрос внести таблицы **Заказы** и **Клиенты**. Связать таблицы по полю **КодКлиента**. Выбрать поля **Название** (из таблицы **Клиенты**), **КодЗаказа**, **ДатаРазмещения** (из таблицы **Заказы**). Меню «**Вид**» _ пункт «**Групповые операции**». В строке «**Групповая операция**» в запросе для поля **КодЗаказа** выбрать операцию **Count**, а для поля **ДатаРазмещения** выбрать **Условие**. В строку «**Условие отбора**» для поля **ДатаРазмещения** внести >=#01/01/1995#. В строке «**Вывод на экран**» для поля **ДатаРазмещения** убрать галочку. Просмотреть запрос в режиме таблицы. Сохранить запрос под именем **ЧислоЗаказов**.

Запрос в MS Access является объектом, который сохраняется в файле базы данных и может многократно повторяться. Предыдущие запросы содержали конкретные значения полей в строке «**Условие отбора**». Если требуется повторить такой запрос с другими значениями в условиях отбора, его нужно открыть в режиме конструктора, изменить условие и выполнить. Чтобы не делать многократно этих операций, можно создать запрос с параметрами. При выполнении такого запроса выдается диалоговое окно **Введите значение параметра**, в котором пользователь может ввести конкретное значение и затем получить желаемый результат.

 19. Запросы с параметрами.

Создать копию запроса **ЧислоЗаказов.** Открыть новый запрос в режиме конструктора. В строку «**Условие отбора**» для поля **ДатаРазмещения** внести

=[С какой даты заказ]

Просмотреть запрос в режиме таблицы. Ввести дату 13 марта 1995.

В режиме конструктора снять с запроса признак «**Групповые операции**» («**Вид**» _ пункт «**Групповые операции**»). В строке «**Вывод на экран**» для поля **ДатаРазмещения** поставить галочку. Просмотреть запрос в режиме таблицы. Ввести дату 13 марта 1995.

В строку «**Условие отбора**» для поля **ДатаРазмещения** внести

between [ДатаНачала] **and** [ДатаКонца]

Меню «**Запрос**» _ пункт «**Параметры**». В столбец «**Параметр**» ввести **ДатаКонца**, выбрать тип «**Дата/время**», **ДатаНачала**, выбрать тип «**Дата/время**». Закрыть окно параметры запроса. Просмотреть запрос в режиме таблицы.

В окне «**Параметры запроса**» поменять местами порядок параметров. Просмотреть запрос в режиме таблицы.

 20. Создать запрос, который позволяет узнать «Какие клиенты сделали заказы (каждый клиент должен попасть в запрос 1 раз) ».

В новый запрос внести таблицы **Заказы** и **Клиенты**. Связать таблицы по полю **КодКлиента**. Выбрать поле **Название** (из таблицы **Клиенты**). Просмотреть запрос в режиме таблицы. Сколько раз клиенты встречаются в запросе? Переключиться в режим конструктора. Открыть окно «**Свойства запроса**» (меню «**Вид**»_ пункт «**Свойства**»). В строке «**Уникальные значения**» свойств поля выбрать «**Да**». Просмотреть запрос в режиме таблицы.

Создание запросов-изменений

- **Запрос на создание таблицы** создает новую таблицу полностью или частично копируя структуру и данные из одной или нескольких таблиц.

 1. Создать таблицу поставщиков приправы.

В новый запрос добавить таблицу **Товары** и **Поставщики**, выбрать поля **Название** (из таблицы **Поставщики**) и **КодТипа** (из таблицы **Товары**). Просмотреть запрос в режиме таблицы. Поле **КодТипа** (числовое), но имеет подстановочное значение в таблице **Товары**. Чтобы узнать число, которое соответствует приправам, откройте таблицу **Типы** и узнайте это значение или в запросе создайте новое поле, в котором напишите [КодТипа]+0. Введите в строку «**Условие отбора**» для поля **КодТипа** соответствующее числовое значение. Просмотреть запрос в режиме таблицы. Убедились, что в запрос попали приправы? Избавьтесь от повторяющихся записей («**Свойства запроса**», «**Уникальные записи**»).

Меню «**Запрос**» _ пункт «**Создание таблицы**» (рис. 21). Далее введите имя новой таблицы - **ПоставщикиПриправ**.

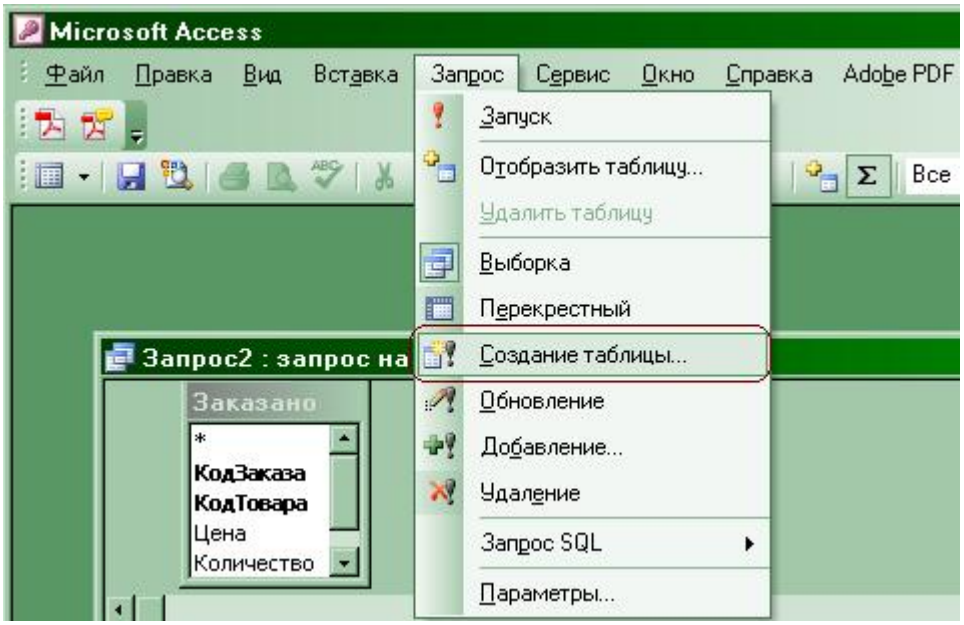


Рис. 21. Создание запросов-изменений

Чтобы запустить запрос: открыть запрос по кнопке «**Открыть**» или в режиме конструктора меню «**Запрос**» _ пункт «**Запуск**» (рис. 21). Опробовать оба варианта. Просмотреть таблицу **ПоставщикиПриправ**. Обратите внимание на значок запроса.

- **Запрос на удаление** - удаляет записи из одной или нескольких таблиц.

1. Из таблицы **ТоварыНовые** удалить все рыбопродукты.

Сначала создайте таблицу **ТоварыНовые** из таблицы **Товары** (выбрать все поля). В запрос добавить таблицу **ТоварыНовые**. Выбрать поле **КодТипа**. В строку «**Условие отбора**» поля **КодТипа** внести код, соответствующий рыбопродуктам. Просмотреть запрос в режиме таблицы. Меню «**Запрос**» _ пункт «**Удаление**».

Чтобы запустить запрос: открыть запрос по кнопке «**Открыть**» или в режиме конструктора «**Запрос**» _ пункт «**Запуск**» (рис. 21). Просмотреть таблицу **ТоварыНовые**, правильно удалили записи?

2. Создать запрос на удаление из таблицы **Заказы** всех заказов, у которых город получателя **Берн**. Подсчитать сколько записей будет удалено из таблицы **Заказы** и сколько записей при этом будет удалено из таблицы **Заказано**.

- **Запрос на добавление записей** добавляет группу записей из одной или нескольких таблиц в одну или несколько других таблиц.

1. В таблицу **ПоставщикиПриправ** добавить поставщиков кондитерских изделий.

Открыть таблицу **ПоставщикиПриправ**, запомнить имена полей этой таблицы. В новый запрос добавить таблицы **Поставщики** и **Товары**. Выбрать поля **КодТипа** (из таблицы **Товары**) и **Название** (из таблицы **Поставщики**). Меню «**Запрос**» _ пункт «**Добавление**». Выбрать таблицу **ПоставщикиПриправ**. Заполнить строку «**Условие отбора**» поля **КодТипа** для кондитерских изделий. Просмотреть запрос в режиме таблицы. Задать в свойствах запроса «**Уникальные записи**». Просмотреть запрос в режиме таблицы.

Чтобы запустить запрос: открыть запрос по кнопке «**Открыть**» или в режиме конструктора «**Запрос**» _ пункт «**Запуск**» (рис. 21). Просмотреть таблицу **ПоставщикиПриправ**.

- **Запрос на обновление записей** изменяет данные в группе записей.

 1. В таблице **ТоварыНовые** провести деноминацию цен (значения в поле **Цена** разделить на 1000).

В новый запрос добавить таблицу **ТоварыНовые**. Выбрать поле **Цена**. В режиме таблицы для запроса запомнить цену любого товара.
Меню «**Запрос**» _ пункт «**Обновление**». В строке «**Обновление**» в поля **Цена** ввести [Цена]/1000.

Чтобы запустить запрос: открыть запрос по кнопке «**Открыть**» или в режиме конструктора «**Запрос**» _ пункт «**Запуск**» (рис. 21). Преобразовать запрос в запрос на выборку и просмотреть поле **Цена** для товара, который запоминали.

- **Перекрестные запросы** — это запросы, в которых происходит статистическая обработка данных, результаты которой выводятся в виде таблицы.

 1. В новый запрос добавить таблицу **Товары**. Выбрать поля **КодТипа**, **Марка**, **НаСкладе**. Добавить новое поле **Итого:НаСкладе**. Меню «**Запрос**» _ пункт «**Перекрестный**». Строки полей заполнить следующим образом:

	КодТипа	**Итого**	**Марка**	**НаСкладе**
Групповая операци:	группировка	Sum	группировка	Sum
Перекрестная таблица:	заголоки строк	заголоки строк	заголоки столбцов	Значение

Просмотреть запрос в режиме таблицы.

Чтобы запустить запрос: открыть запрос по кнопке «**Открыть**» или в режиме конструктора «**Запрос**» _ пункт «**Запуск**» (рис. 21). Сохранить запрос под именем **НаСкладе**.

Аналогичный запрос можно создать мастером запросов (кнопка «**Создать**», «**Перекрестный запрос**»).

 2. Создать из запроса **НаСкладе** новую таблицу (использовать этот запрос в качестве основы другого запроса).

Лабораторная работа №3

Работа с формами

Цель: Приобрести умения и навыки при работе с формами. Научиться создавать формы с помощью конструктора форм и с помощью мастера форм. Научиться форматировать структуру формы.

Форма используется для ввода данных в таблицу и для просмотра в заданном формате данных из таблицы или запроса. С ее помощью можно также запустить на выполнение макрос или процедуру.

Большая часть данных, представленных в форме берется из таблицы или запроса, кроме того форма может содержать информацию, не связанную ни с таблицей ни с запросом.

Форма, которая не связана ни с какой таблицей и ни с каким запросом называется **несвязанной**.

Все данные в форме хранятся внутри элементов управления. Элементы управления - это объекты, размещенные в форме и предназначенные для изображения данных, выполнения операций или просто для красоты.

Связанный элемент управления это элемент управления, источником данных которого является поле таблицы или запроса.

Несвязанный элемент управления это элемент управления, источник данных которого не определен.

Вычисляемый элемент управления это элемент управления, источником данных которого является выражение (а не поле). Например,
=[Цена]*0,75

Формы в элементах управления «**Поле**» наследуют свойства базовых таблиц и запросов.

Стандартные режимы окна формы:

- простая (для показа 1 записи),
- ленточная (для изображения нескольких записей),
- таблица (записи в табличном формате, используется для подчиненных форм).

Задание: В Help найти и прочитать информацию про формы, режимы формы.

1. Из базы **Борей.mdb** заимпортировать таблицу **Поставщики** (если ее у вас нет).

2. С помощью запроса на создание таблицы создать таблицу **ПоставщикиКопия**.

Создание формы с помощью мастера форм

Если на вашем компьютере не установлен мастер создания форм, то создайте форму с помощью конструктора (далее), но выполните все задания этого раздела, начиная с 6-го.

1. Чтобы создать новую форму с помощью мастера форм следует выбрать объект «**Формы**» и щелкнуть по кнопке «**Создать**» (рис.22). В открывшемся окне «**Новая форма**» выбрать «**Мастер форм**».

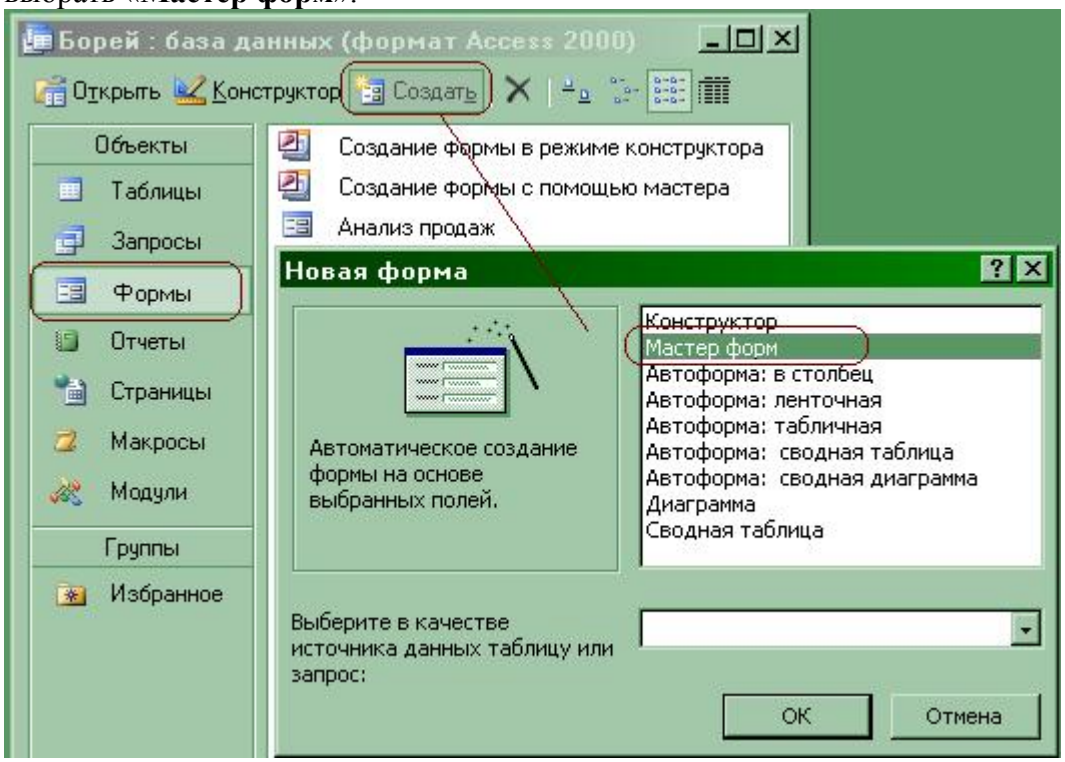


Рис. 22. Создание формы

Далее следуем за мастером форм.

2.В качестве источника данных выбрать таблицу **ПоставщикиКопия**, кнопка «**ОК**».

3. Выбрать все поля кнопка «>>», кнопка «**Далее**».

4. Внешний вид формы: «**В один столбец**».

5. Выбрать любой стиль для подписей, кнопка «**Далее**».

6. Задать имя формы **ФормаПоставщикиКопия** (можно задать любое имя), кнопка «**Готово**». Мастер форм создал форму в режиме «**простая форма**». Источником данных формы является таблица **ПоставщикиКопия**.

7. Поперемещаться по записям. Сколько записей всего в таблице **ПоставщикиКопия**? На какой записи по номеру вы находитесь? Переместиться на новую запись. Эти действия проделать с помощью мыши, используя навигатор по записям (внизу формы), или меню «**Правка**» _ пункт «**Перейти**» _ различные варианты перехода.

8. Найти область выделения записи. Выделить текущую запись мышью или меню «**Правка**» _ пункт «**Выделить запись**».

9. Удалить текущую запись (выделить запись, клавиша «**Delete**» или меню «**Правка**» _ пункт «**Удалить запись**».

10. Переключиться в режим конструктора формы (меню «**Вид**» _ пункт «**Конструктор**» или по кнопке  на панели инструментов). Найти окно свойств формы (меню «**Вид**» _ пункт «**Свойства**»). Найти в свойствах формы ссылку на источник записей (закладка «**Данные**» свойство «**Источник записей**»). Для выбора свойств формы, щелкнуть мышью на квадратике слева от горизонтальной линейки на форме.

11. Выбрать поле для адреса. В свойствах поля найти свойство, которое отражает какие данные связаны с выбранным полем (вкладка «**Данные**», свойство «**Данные**»). Из списка доступных в форме данных выбрать **Адрес**. Изменить имя текущего поля (вкладка «**Другие**», свойство «**Имя**», изменить имя на **Адрес**). Изменить надпись для поля адрес (выбрать надпись, которую хотим менять, вкладка «**Макет**», свойство «**Подпись**», изменить значение). Просмотреть форму в режиме формы. Аналогичные действия проделать для всех полей, у которых в полях стоит #?Имя.

12. Сделать источником записей формы таблицу **Поставщики**. Просмотреть форму в режиме формы. Попробовать удалить какую-нибудь запись из таблицы **Поставщики** через форму. Нацелить форму снова на таблицу **ПоставщикиКопия**.

13. Добавить новую запись в таблицу.

14. Изменить подпись у формы (свойства формы, вкладка «**Макет**», свойство «**Подпись**» на **Поставщики товаров**). Просмотреть форму в режиме формы.

15. Поле **КодПоставщика** сделать невидимым полем. В режиме конструктора выбрать поле (использовать клавишу **Shift** для нескольких полей). Свойства поля, вкладка «**Макет**», свойство «**Вывод на экран**» выбрать значение **Нет**. Просмотреть форму в режиме формы.

16. Сделать ширину всех полей одинаковой 5 см. В режиме конструктора выбрать все поля. Свойства поля, вкладка «**Макет**», свойство «**Ширина**» написать 5.

17. Расположить данные в 2 столбца. Выбрать поля **Область** и **Индекс** (использовать **Shift**). Подвести указатель мыши к выбранным полям, и когда указатель примет вид ладошки, нажать левую кнопку мыши и, не отпуская мышь, перенести поля на новое место (если что-то не так, меню «**Правка**» _ пункт «**Отменить сдвиг**»). Подвести указатель мыши по вертикальной линейке к полю **Страна** и когда указатель примет вид стрелки, нажать на левую кнопку мыши и, не отпуская мышь сдвинуть указатель мыши вниз по линейке до поля **Факс**. Теперь переместить блок выбранных полей на новое место.

18. Запретить удаление записей через форму из таблицы **ПоставщикиКопия**. Изменить свойства формы:
вкладка «**Макет**», «**Область выделения**», значение **Нет**
вкладка «**Данные**», «**Разрешить удаление**» , значение **Нет**.
Просмотреть форму в режиме формы. Попробовать удалить запись через форму.

Создание формы с помощью конструктора

1. Создать новую форму с помощью конструктора. Чтобы создать новую форму в режиме конструктора следует выбрать объект «**Формы**» и щелкнуть по кнопке «**Создать**» (рис.22). В открывшемся окне «**Новая форма**» выбрать «Конструктор». В качестве источника данных для формы выбрать таблицу **ПоставщикиКопия**.

На форму можно добавить визуальные элементы управления с «**Панели элементов**». Чтобы открыть «**Панель элементов**» следует в меню «**Вид**» выбрать пункт «**Панель элементов**» или по кнопке  на панели инструментов.

Рис. 23. Панель элементов

2. На панели элементов выбрать элемент «**Поле**» (1 раз щелкнуть левой кнопкой мыши на элементе). Перенести «**Поле**» на форму. В качестве источника данных для поля выбрать **КодПоставщика**. Поменять значения свойств «**Имя**» для поля и «**Подпись**» для элемента «**Надпись**». Просмотреть форму в режиме формы.

3.В режиме конструктора копировать поле с кодом поставщика во внутренний буфер (выбрать поле, меню «**Правка**» _ пункт «**Копировать**»). Вставить поле на форму из внутреннего буфера. В качестве источника данных для нового поля выбрать **Название**. Поменять значения свойств «**Имя**» для поля и «**Подпись**» для элемента «**Надпись**». Просмотреть форму в режиме формы.

4.Открыть форму **ФормаПоставщикиКопия** в режиме конструктора. Выбрать все поля, кроме полей **КодПоставщика** и **Название**. Копировать эти поля во внутренний буфер. Переключиться на новую форму и вставить в нее содержимое буфера. Разместить поля так, чтобы они друг друга не перекрывали. Просмотреть источники данных для перенесенных

полей. Просмотреть форму в режиме формы. Сохранить форму под именем **ФормаПоставщикиКопия1**.

 5.Чем отличаются формы **ФормаПоставщикиКопия** и **ФормаПоставщикиКопия1**? (возможно расположением полей и форма **ФормаПоставщикиКопия1** позволяет делать удаление записей).

 6. В форме **ФормаПоставщикиКопия1** оставить 5 первых полей, остальные удалить. Поджать форму так, чтобы она вмещала поля. Просмотреть форму в режиме формы.

 7.В свойствах формы изменить свойство «**Режим по умолчанию**» (вкладка «**Макет**») на «Л**енточные формы**». Просмотреть форму в режиме формы. Чем данный режим отличается от режима «**Простая форма**»? (Можно просмотреть несколько записей).

 8. В свойствах формы изменить свойство «**Режим по умолчанию**» на «**Режим таблицы**». Просмотреть форму в режиме формы. Чем данный режим отличается от режима «**Простая форма**» и «Л**енточные формы**»? (данные представлены в табличном виде, а не по отдельным полям).

Лабораторная работа №4

Открытие формы из другой формы по кнопке

Цель: Приобрести умения и навыки при работе с формами. Научиться разными способами открывать формы.

 1. Открыть форму **ФормаПоставщикиКопия** (или любую другую) из другой формы по кнопке.

1.1. Создать с помощью конструктора форму **Запуск**.

1.2. На панели элементов выбрать элемент **«Кнопка»** и перенести ее на форму. Начинает работать мастер создания кнопок. В списке **«Категории»** выбрать «**Работа с формой**», в списке «**Действия**» выбрать «**Открытие формы**», кнопка «**Далее>**». Выбрать в списке форму **ФормаПоставщикиКопия** (или другую), которая будет вызываться при нажатии данной кнопки, кнопка «**Далее>**». Выбрать «**Открыть форму и показать все записи**», кнопка «**Далее>**». Выбрать «**Текст**» и написать **Открытие через процедуру формы**, кнопка «**Далее>**». Имя кнопки оставить **Кнопка0,** кнопка «**Готово**».

1.3. В макете свойств формы убрать полосы прокрутки (свойство «**Полосы прокрутки**»), область выделения (свойство «**Область выделения**»), поле номера записи (свойство «**Кнопки перехода**»), разделительные линии (свойство «**Разделительные линии**»).

1.4. Изменить подпись к форме на **Главная форма для запуска** (свойство «**Подпись**»). Просмотреть форму в режиме формы.

1.5. Просмотреть процедуру обработки кнопки (режим конструктора для формы, свойства для кнопки, вкладка «**События**», кнопка «**...**» справа от процедура обработки события в строке «**Нажатие кнопки**». Какая команда в коде открывает форму?

1.6. Прочитать Help по команде **Docmd** (F1 на имени команды). Прочитать Help по методу **OpenForm**. Разобраться с аргументами.

Метод **OpenForm** выполняет макрокоманду **ОткрытьФорму (OpenForm)** в программе Visual Basic.

Синтаксис

DoCmd.OpenForm имяФормы [, режим] [, имяФильтра] [, условиеWhere] [, режимДанных] [, режимОкна] [, аргументыОткрытия]

Параметры (аргументы) в квадратных скобках являются необязательными. Метод OpenForm использует следующие аргументы.

Аргумент	Описание
имяФормы	Строковое выражение, представляющее допустимое имя формы в текущей базе данных.
режим	Одна из следующих встроенных констант **acDesign** (конструктор) **acFormDS** (таблица) **acNormal** (форма) - задает открытие формы в режиме формы (значение по умолчанию). **acPreview** (просмотр)
имяФильтра	Строковое выражение, представляющее допустимое имя запроса в текущей базе данных.
условиеWhere	Строковое выражение, представляющее допустимое предложение SQL WHERE без ключевого слова WHERE.
режимДанных	Одна из следующих встроенных констант: acFormAdd (добавление) acFormEdit (изменение) acFormPropertySettings (значения свойств) acFormReadOnly (только чтение) Если оставить данный аргумент пустым (предполагается значение по умолчанию acFormPropertySettings), Microsoft Access открывает форму в режиме данных, который определяется значениями свойств Разрешить изменение (AllowEdits), Разрешить удаление (AllowDeletions), Разрешить добавление (AllowAdditions) и Ввод данных (DataEntry).
режимОкна	Одна из следующих встроенных констант: acDialog (окно диалога) acHidden (невидимое) acIcon (значок) acWindowNormal (обычное) Если оставить данный аргумент пустым, подразумевается значение по умолчанию acWindowNormal.
аргументыОткрытия	Строковое выражение, определяющее значение свойства формы OpenArgs. В дальнейшем это значение может быть использовано в программе в модуле формы, например, в процедуре обработки события Открытие (Open). Ссылки на свойство OpenArgs допускаются также в макросах и выражениях.

Необязательный аргумент посреди списка аргументов разрешается пропустить, однако, при этом необходимо ввести запятую, представляющую пропущенный аргумент. Если пустыми оставлены последние аргументы, вводить запятые вслед за последним указанным аргументом не требуется.

В следующем примере в форме «Сотрудники», открывающейся в режиме формы, выводятся только записи со значением «Иванов» в поле «Фамилия». Допускается изменение выводящихся записей и добавление новых.

```
DoCmd.OpenForm "Сотрудники", , ,"Фамилия = 'Иванов'"
```

1.7. Запомнить имя кнопки (свойство «**Имя**», не подпись!). Изменить имя кнопки на **OpenProc** (Свойства кнопки, вкладка «**Другие**», свойство «**Имя**»). В режиме формы нажать на кнопку. Что происходит? Почему? В режиме конструктора для формы

просмотреть процедуру обработки события «**Нажатие кнопки**». Что содержит процедура **OpenProc_Click**? Найти процедуру **Кнопка0_Click**, изменить ее имя на **OpenProc_Click**,

Сохранить, откомпилировать (меню «**Debug**», «**Compile** » или по кнопке на панели инструментов). Опробовать работу кнопки. Открывает форму?

2. Открыть форму **ФормаПоставщикиКопия** (или любую другую) из формы **Запуск** по новой кнопке в **режиме добавления новых записей** в таблицу **ПоставщикиКопия**.

2.1. Копировать ранее созданную кнопку, вставить ее на форму Запуск. Подпись к кнопке **Открытие через процедуру формы для добавления**.

2.2. Дать кнопке имя **OpenProcAdd** (не подпись!). Создать для этой кнопки процедуру обработки события «**Нажатие кнопки**», копировав содержимое процедуры **OpenProc.** Изменить аргументы в **Docmd.OpenForm** так, чтобы через форму **ФормаПоставщикиКопия** можно было только добавлять новые записи. Откомпилировать процедуру. Опробовать добавление новой записи.

2.3. Проверить, что записи добавились (кнопка **Открытие через процедуру формы**)

2.4. Сделать так, чтобы по кнопке **Открытие через процедуру формы** можно было только читать записи (Изменить аргументы в **Docmd.OpenForm**). Не забыть откомпилировать! Опробовать.

2.5. Сделать так, чтобы по кнопке **Открытие через процедуру формы** форма открывалась в **режиме таблицы** (Изменить аргументы в **Docmd.OpenForm**). Не забыть откомпилировать! Опробовать.

2.6. Используя форму **Запуск**, добавить новую запись в таблицу **ПоставщикиКопия**, просмотреть эту и другие записи таблицы, изменить последнюю добавленную запись. Что происходит? Почему?

2.7. Создать кнопку **Открытие через процедуру формы для редактирования**, которая бы позволяла редактировать записи в форме **ФормаПоставщикиКопия**.

2.8. Сделать так, чтобы по кнопке **Открытие через процедуру формы** можно было просматривать только поставщиков из Франции (Изменить аргументы в **Docmd.OpenForm**, Help и примеры)

3. В форму **Запуск** добавить новую кнопку **Открытие из макроса**, которая будет открывать форму **ФормаПоставщикиКопия** из макроса.

Макрос представляет собой набор из одной или нескольких макрокоманд (описание стандартных действий, которые нужно выполнить в ответ на определенное событие), каждая из которых выполняет определенное действие - например, открывает форму или отчет. Макросы применяются для автоматизации часто выполняемых операций.

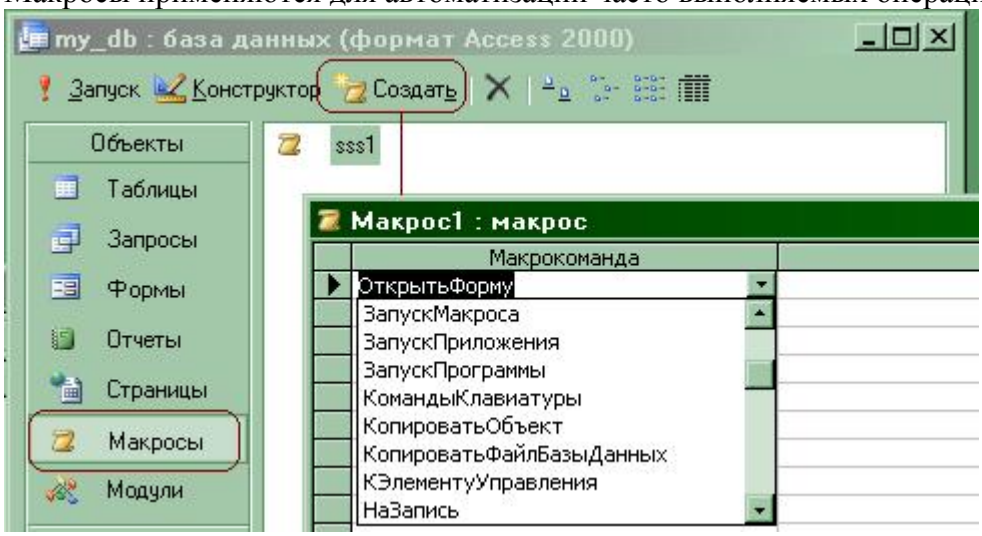


Рис. 24. Создание макроса

3.1. Добавить новую кнопку **Открытие из макроса**. На событие «**Нажатие кнопки**» создать макрос. Выбрать объект «**Макросы**», кнопка «**Создать**», выбрать в списке имя макрокоманды «**Открыть форму**», задать аргументы к макрокоманде, дать макросу имя **sss1**. Опробовать действие кнопки.

3.2. Создать кнопки, аналогичные кнопкам **Открытие через процедуру формы.....,** но открывающие форму через макросы (**sss1, sss2, sss3**).

4. Создать кнопки, которые проделают действия аналогичные предыдущим, но через процедуры, которые будут находиться в модуле **MyMod**.

Модуль — это программа, написанная на языке Visual Basic for Applications (VBA). Использование модулей позволяет автоматизировать выполнение сложных действий, которые нельзя описать с помощью макросов.

Рис. 24. Создание модуля

4.1. Создать новый модуль **MyMod** (Выбрать объект «**Модули**», кнопка «**Создать**»).

4.2. Вставить в модуль новую процедуру **but1** (меню «**Insert**» (Вставка), «**Procedure**» (Процедура)), в которую перенести содержимое процедуры **OpenProc_Click** из формы **Запуск**.

4.3. Создать новую кнопку **Открытие из модуля**. В процедуре обработки события «**Нажатие кнопки**» сделать вызов процедуры **but1**. Не забыть откомпилировать. Опробовать действие кнопки.

В результате вы должны получить форму **Запуск** с 9 кнопками, которые запускают открытие одной и той же формы, но в разных режимах для формы, в разных режимах работы с данными и разными способами открывают форму.

Общие сведения о подчиненных формах

Подчиненная форма - это форма, находящаяся внутри другой формы. Первичная форма называется главной формой, а форма внутри формы называется подчиненной формой. Вместе главная и подчиненная форма называются составной формой.

Подчиненная форма удобна для вывода данных из таблиц или запросов, связанных отношением "один-ко-многим".

Главная форма и подчиненная форма связаны таким образом, что в подчиненной форме выводятся только те записи, которые связаны с текущей записью в главной форме.
При использовании формы с подчиненной формой для ввода новых записей текущая запись в главной форме сохраняется при входе в подчиненную форму. Это гарантирует, что записи из таблицы на стороне "многие" будут иметь связанную запись в таблице на стороне "один". Это также автоматически сохраняет каждую запись, добавляемую в подчиненную форму.

Подчиненная форма может быть выведена в режиме таблицы, или она может быть выведена как простая или ленточная форма. Главная форма может быть выведена **только как простая форма**.

Главная форма может содержать любое число подчиненных форм, если каждая подчиненная форма помещается в главную форму. Имеется также возможность создавать подчиненные формы двух уровней вложенности. Это означает, что можно иметь подчиненную форму внутри главной формы, а другую подчиненную форму внутри этой подчиненной формы.

1. Создать форму **ЗаказыФирмы**, источником записей которой служит таблица **Заказы**, которая будет содержать подчиненную форму **ЗаказаноПоЗаказу**.

1.1. С помощью мастера форм создать форму **ЗаказыФирмы** (если мастер форм не установлен, создайте форму с помощью конструктора). Эта форма будет главной.

1.2. На панели элементов выбрать элемент «**Подчиненная форма/отчет**» и перенести ее на вашу форму. Когда запустится мастер создания подчиненных форм, выбрать «**Имеющиеся Таблицы и Запросы**», кнопка «**Далее>**», в списке таблиц выбрать **Заказано**, кнопка «**Далее>**». Выбрать «**Самостоятельное определение**», отобрать все поля, завязать формы по полю **КодЗаказа** (выбрать соответствующие поля из списков), кнопка «**Далее>**». Подчиненной форме дать имя **ЗаказаноПоЗаказу**, кнопка «**Готово**» (рис. 25). Просмотреть форму **ЗаказыФирмы** в режиме формы.

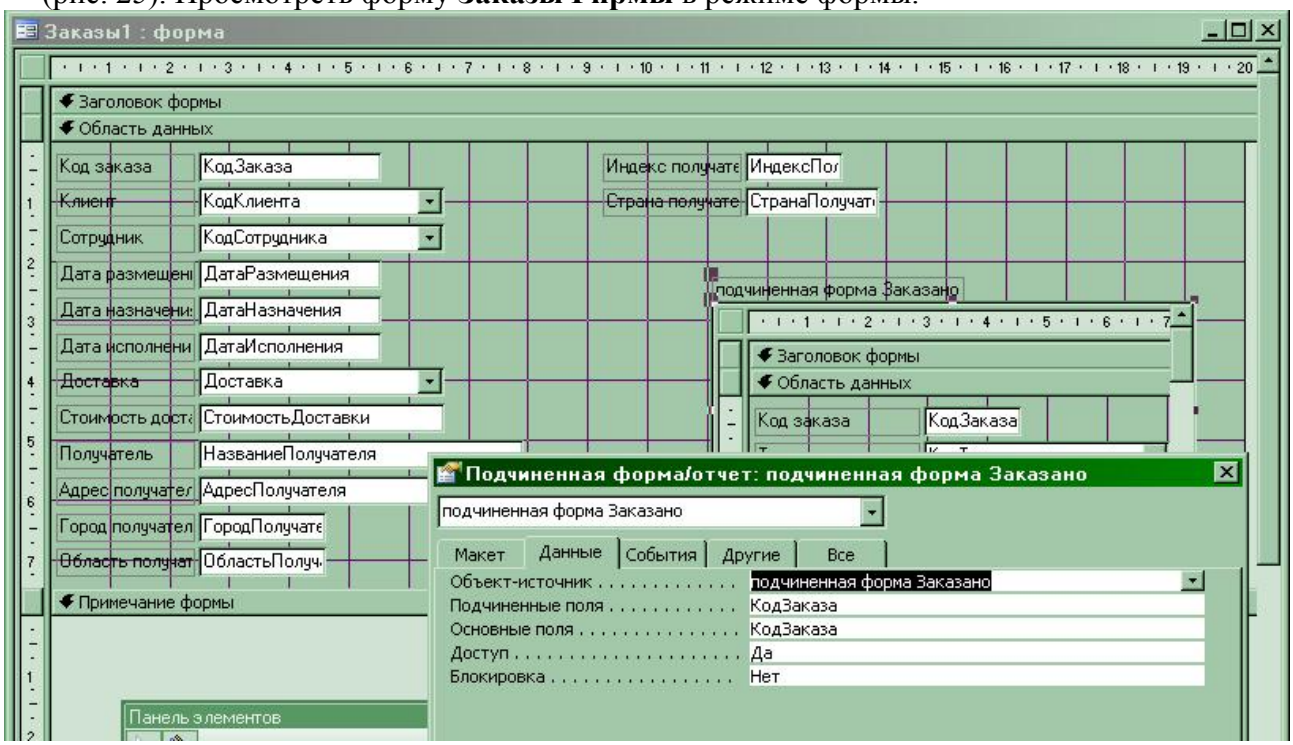


Рис. 25. Мастер создания подчиненных форм установил связь между главной и подчиненной формами

1.3. Изменить размер подчиненной формы так, чтобы было удобно просматривать записи внутри подчиненной формы.

1.4. Поперемещаться по заказам в главной форме. Что происходит внутри подчиненной формы?

1.5. Какие товары заказаны в каждом заказе? Удобно работать с кодами? Как в поле **КодТовара** получить наименование товара? Изучить каким образом сделана подстановка в поле **КодТовара** таблицы **Заказано**. Проделать то же самое для подчиненной формы **ЗаказаноПоЗаказу** (произвести соответствующие изменения в форме **ЗаказаноПоЗаказу** в режиме конструктора). Добиться, чтобы в поле **КодТовара** попадала строка названия товара. Просмотреть форму **ЗаказыФирмы** после этих изменений.

1.6. Создать запрос **ЗаказыНовые,** в который поместить все поля из таблицы **Заказы**. Переименовать поле **КодЗаказа** в **Код**.

1.7. Поменять источник записей формы **ЗаказыФирмы** на запрос **ЗаказыНовые**. Изменить источник данных поля **КодЗаказа** формы **ЗаказыФирмы**. Просмотреть форму **ЗаказыФирмы** в режиме формы. Поля **КодЗаказа** в форме уже нет. Почему просят ввести значение параметра **КодЗаказа**? Почему при переходе на новый заказ нет изменений в подчиненной форме?

1.8. Открыть форму **ЗаказыФирмы** в режиме конструктора. Выбрать подчиненную форму и просмотреть ее свойства по данным.

Правильно ли указано свойство «**Объект-источник**» (имя подчиненной формы)?

Правильно ли выбраны «**Подчиненные поля**» (поле(я), по которым организована связь в подчиненной форме)?

Правильно ли указано свойство «**Основные поля**» (поле(я), по которым организована связь в главной форме)?

Сделать необходимые изменения. Просмотреть форму.

1.9. Подчиненную форму в главной форме вывести в режиме таблицы, в режиме ленточной формы, в режиме простой формы.

Все открытые формы MS Access автоматически помещает в коллекцию **Application.Forms**. Каждая форма представляется в виде объекта **Form**. Объект **Form** - нормальный объект, который имеет свойства, методы и события. Различные варианты обращения к форме через коллекцию форм показаны в табл.2.

Таблица 2.

Синтаксис	Пример	Описание
Forms!*formname*	Forms!OrderForm	Обращение по имени формы.
Forms![*form name***]**	Forms![Order Form]	Обращение по имени формы, когда в имени формы содержится пробел.
Forms("*formname***")**	Forms("OrderForm")	Обращение по имени формы.
Forms(*index***)**	Forms(0)	Обращение по номеру формы в коллекции форм. 0 первая форма в коллекции

Чтобы вызвать свойство или метод внутри модуля формы следует написать
me.ИмяСвойства
me.ИмяМетода

Все элементы управления на форме помещаются в коллекцию **Controls**, доступ к которой осуществляется через свойство **Controls** объекта **Form**. Различные варианты обращения к элементам управления на форме показаны в табл.3.

Таблица 3.

Пример	Описание
Forms!OrderForm!NewData	Обращение по имени формы и имени элемента управления.
Me!NewData	Обращение по имени элемента управления внутри модуля формы.
Me![New Data]	Обращение по имени элемента управления внутри модуля формы, когда в имени содержится пробел.
Me("NewData")	Обращение по имени элемента управления внутри модуля формы.
Me(0)	Обращение к первому элементу коллекции.

Обращение к элементу управления на форме через коллекцию элементов управления представлено в табл. 4.

Таблица 4.

Пример	Описание
Forms!OrderForm.Controls!NewData	Обращение по имени формы и имени элемента управления.
Me.Controls!NewData	Обращение по имени элемента управления внутри модуля формы.
Me.Controls![New Data]	Обращение по имени элемента управления внутри модуля формы, когда в имени содержится пробел.
Me.Controls("NewData")	Обращение по имени элемента управления внутри модуля формы.
`Me.Controls(0)`	Обращение к первому элементу коллекции.

Обращение к свойствам и элементам управления форм

Mainform – имя главной формы
Subform1 –имя подчиненной формы первого уровня
Subform2 –имя подчиненной формы второго уровня
ControlName – имя элемента управления формы

Синтаксис обращения зависит от того, из какой формы на форму какого уровня вложенности ссылаетесь.

Ссылка на источник данных формы (RecordSource)

Из Mainform на Mainform – **Me.RecordSource**
Из Mainform на Subform1– **Me!Subform1.Form.RecordSource**
Из Mainform на Subform2 – **Me!Subform1.Form!Subform2.Form.RecordSource**

Из Subform1 на Mainform – **Me.Parent.RecordSource**
Из Subform1 на Subform1 – **Me.RecordSource**
Из Subform1 на Subform2 –**Me!Subform2.Form.RecordSource**

Из Subform2 на Mainform – **Me.Parent.Parent.RecordSource**
Из Subform2 на Subform1 – **Me.Parent.RecordSource**
Из Subform2 на Subform2 – **Me.RecordSource**

Из внешней формы на Mainform – **Forms!Mainform.RecordSource**
Из внешней формы на Subform1 – **Forms!Mainform!Subform1.Form.RecordSource**
Из внешней формы на Subform2 –
Forms!Mainform!Subform1.Form!Subform2.Form.RecordSource

Ссылка на элементы управления (Controls)

Из Mainform на Mainform – **Me!ControlName**
Из Mainform на Subform1– **Me!Subform1.Form!ControlName**
Из Mainform на Subform2 – **Me!Subform1.Form!Subform2.Form!ControlName**

Из Subform1 на Mainform – **Me.Parent!ControlName**
Из Subform1 на Subform1 – **Me!ControlName**
Из Subform1 на Subform2 – **Me!Subform2.Form!ControlName**

Из Subform2 на Mainform – **Me.Parent.Parent!ControlName**
Из Subform2 на Subform1 – **Me.Parent!ControlName**

Из Subform2 на Subform2 – **Me!ControlName**

Из внешней формы на Mainform – **Forms!Mainform!ControlName**
Из внешней формы на Subform1 – **Forms!Mainform!Subform1.Form!ControlName**
Из внешней формы на Subform2 –
Forms!Mainform!Subform1.Form!Subform2.Form!ControlName

Управление доступом (Enabled)

Из Mainform на Mainform – **Me!ControlName.Enabled**
Из Mainform на Subform1 – **Me!Subform1.Form!ControlName.Enabled**
Из Mainform на Subform2 – **Me!Subform1.Form!Subform2.Form!ControlName.Enabled**

Из Subform1 на Mainform – **Me.Parent!ControlName.Enabled**
Из Subform1 на Subform1 – **Me!ControlName.Enabled**
Из Subform1 на Subform2 – **Me!Subform2.Form!ControlName.Enabled**

Из Subform2 на Mainform – **Me.Parent.Parent!ControlName.Enabled**
Из Subform2 на Subform1 – **Me.Parent!ControlName.Enabled**
Из Subform2 на Subform2 – **Me!ControlName.Enabled**

Из внешней формы на Mainform – **Forms!Mainform!ControlName.Enabled**
Из внешней формы на Subform1 – **Forms!Mainform!Subform1.Form!ControlName.Enabled**
Из внешней формы на Subform2 –
Forms!Mainform!Subform1.Form!Subform2.Form!ControlName.Enabled

 2. Создать форму **ПросмотрЗаказов**, в которой данные по заказам будут просматриваться в режиме таблицы.

1 вариант
2.1. Создать форму **Подч1**, которая будет использована в качестве подчиненной формы. В форме **Подч1** отображать заказы в табличном виде.
2.2. Создать форму **Подч2**, которая будет использована в качестве подчиненной формы. В форме **Подч2** отображать товары по заказам в табличном виде. Правильно определите таблицы, которые будут входить в состав запроса, который будет источником данных для подчиненной формы.
2.3. Создайте главную форму **Главная1,** в которую поместите две подчиненные формы **Подч1** и **Подч2 (**использовать элемент «**Подчиненная форма/отчет**»**)**. Форма **Главная1** не должна иметь источника записей.
2.4. У подчиненной формы **Подч1** указать объект-источник (имя реально существующей формы, которая будет объектом-источником для элемента «**Подчиненная форма/отчет**»). Подчиненных полей и основных полей у нее нет.
2.5. Подчиненная форма **Подч2** является подчиненной по отношению к форме **Подч1.** У формы **Подч2** указать объект-источник. Указать подчиненные поле(я), т.е. поле из формы **Подч2.**Обращение к основному полю написать в виде
2.6.
[Подч1**].Form![**имя поля из **Подч1** по которому связаны формы **Подч1** и **Подч2]**

Использовать Help для изучения всего нового.
2.6. Просмотреть форму **Главная1** в режиме формы. Поперемещаться по заказам. Происходит изменение записей в форме **Подч2** при перемещении по заказам в **Подч1**?

Создать для формы **Подч1** процедуру обработки события «**Текущая запись**» и в ней написать **me.Parent![Подч2].Requery**. Откомпилировать процедуру. Использовать Help для

изучения всего нового **(Form** object (Объект **Form**) справочной системы Microsoft **Access).** Просмотреть форму **Главная1** в режиме формы. Поперемещаться по заказам. Что происходит?

2 вариант

2.7. Создать форму **Главная2** копировав форму **Главная1.**

2.8. Создать формы **Подч3 и Подч4** копировав формы **Подч1 и Подч2**

2.9. У подчиненной формы **Подч3** (находимся в конструкторе формы **Главная2)** указать объект-источник. Подчиненных полей и основных полей у нее нет.

2.10. Добавить в форму **Главная2** поле, которому дать имя **ww**.

Создать для формы **Подч3** процедуру обработки события. В событии «**Текущая запись**» написать **me.Parent!ww=me!КодЗаказа**. Откомпилировать процедуру. Использовать Help для изучения всего нового.

2.11. У подчиненной формы **Подч4** указать объект-источник. Указать подчиненные поле(я), т.е. поле из формы **Подч4.** Указать основное поле **ww**. Просмотреть форму **Главная2** в режиме формы. Поперемещаться по заказам. Что происходит?

2.12. Поле **ww** сделать невидимым.

Лабораторная работа №5

Работа с записями таблиц через форму

Цель: Приобрести умения и навыки при работе с формами. Научиться работать с записями таблиц через форму.

Работаем с формой **ЗаказыФирмы** (главная форма с подчиненной, главная форма источником записей имеет таблицу **Заказы**, а подчиненная - таблицу **Заказано**). Для дальнейшей работы можно воспользоваться мастерами (если они установлены) или создавать процедуры обработки кнопок вручную.

 1. В форму **ЗаказыФирмы** добавить кнопку **Сохранить1** (имя и подпись кнопки). Процедура обработки этой кнопки должна сохранять текущую запись и форма должна оставаться на этой записи). Использовать метод **DoMenuItem** команды **Docmd** (Help с примерами)

 2. В форму **ЗаказыФирмы** добавить кнопку **СохранитьЗакрыть** (имя и подпись кнопки). Процедура обработки этой кнопки должна сохранять текущую запись и закрыть форму. Воспользоваться тем, что MS Access при закрытии формы сохраняет текущую запись. Использовать метод **Close** команды **Docmd** (Help с примерами)

 3. В форму **ЗаказыФирмы** добавить кнопку **СохранитьНовая** (имя и подпись кнопки). Процедура обработки этой кнопки должна сохранять текущую запись и перейти в форме к созданию новой записи. Воспользоваться тем, что MS Access при переходе в форме к следующей записи сохраняет текущую запись. Использовать метод **GoToRecord** команды **Docmd** (Help с примерами)

 4. В форму **ЗаказыФирмы** добавить кнопку **Отмена** (имя и подпись кнопки). Процедура обработки этой кнопки должна отказаться от вновь введенной (еще не сохраненной) текущей записи. Воспользоваться тем, что MS Access при нажатии «**ESC**» «**ESC**» не сохраняет новую запись. Первый «**ESC**» отменяет значение в поле, а второй «**ESC**» отменяет изменение всей записи. Использовать инструкцию **Sendkeys** (Help с примерами). Свойство «**Отмена**» кнопки установить в «**Да**». Использовать
SendKeys "+{ESC}", True

SendKeys "+{ESC}", True

 5. В форму **ЗаказыФирмы** добавить кнопку **Удалить1** (имя и подпись кнопки). Процедура обработки этой кнопки должна удалить текущую запись и форма должна оставаться открытой). Использовать метод **DoMenuItem** команды **Docmd** (Help с примерами). Удаление происходит в 2 этапа, сначала запись должна быть выделена, а затем удалена.

 6. В форму **ЗаказыФирмы** добавить кнопку **Удалить2** (имя и подпись кнопки). Процедура обработки этой кнопки должна удалить текущую запись из таблицы **Заказы** используя SQL строки.

6.1. Создать запрос на удаление из таблицы **Заказы**, указав в условии для запроса какой-нибудь конкретный заказ, например 10249.

6.2. Переключиться в режим SQL для запроса и копировать строку SQL во внутренний буфер.

6.3. В процедуру обработки кнопки **Удалить2** вставить эту SQL строку, присвоив ее значение переменной sqlstr, т.е. сформировать строковое выражение

sqlstr = "..."

sqlstr = sqlstr + " ... "

6.4. Теперь заменить абсолютное значение для **КодЗаказа** на значение этого поля из формы, преобразовав его к строковому виду **CStr(me!КодЗаказа)**.

6.5. Корректно сформировать строковое выражение для sqlstr, используя операции работы со строками (+, &).

6.6. Использовать метод **RunSql** команды **Docmd** (Help с примерами) для запуска запроса. Откомпилировать процедуру. Опробовать форму. Что происходит при перемещении по записям после удаления записи?

6.7. Использовать метод **Requery** команды **Docmd** (Help с примерами) для перевывода формы на экран после удаления записи (чтобы обновить содержимое источника записи для формы).

6.8. Чтобы при удалении записи по кнопке запретить вывод системных сообщений, использовать метод **SetWarnings** команды **Docmd** (Help с примерами)

Метод **RunCommand** полностью заменил устаревший метод **DoMenuItem**, используемый в предыдущих версиях MS Access, однако мастер создания кнопок активно пользуется устаревшим методом **DoMenuItem**.

Метод **RunCommand** используется для выполнения из программного кода действий, которые Вы обычно выполняете с помощью меню или панелей инструментов. Практически на каждую команду меню MS Access (и контекстные тоже) имеется соответствующая константа для вызова этой команды из кода программы VBA. Из этого следует, что, если Вы видите полезную команду меню или кнопку на панели инструментов и хотите программно использовать действие этой команды (или кнопки), то Вам необходимо лишь в Вашем коде передать соответствующую константу методу **RunCommand**. Действие, вызываемое из программы, будет точно таким же, как если бы Вы вручную выбрали соответствующую команду из меню, контекстного меню (или нажали кнопку на панели инструментов).

Чтобы выполнить соответствующую команду меню и панели инструментов, придерживайтесь такого синтаксиса:

[Application].RunCommand command

или

[DoCmd].RunCommand command

где **command** - это соответствующая команда меню или панели инструментов. Префикс [Application]или [DoCmd] вообще-то, не обязателен:

RunCommand command

Хотя во многих случаях действия метода **RunCommand** и выполнение команд меню похожи, следует различать методы объекта **DoCmd** и методы **RunCommand** (Вообще-то метод RunCommand и сам является методом объекта DoCmd со специальным предназначением - исполнять из программного команды меню MS Access). Наиболее важное различие между ними заключается в том, что, когда Вы используете метод **DoCmd**, Вы можете указать соответствующие аргументы. Когда же Вы используете метод **RunCommand** - Вы просто выполняете определенную команду меню.

Лабораторная работа №6

Создание отчетов

Цель: Приобрести умения и навыки при работе с отчетами. Научиться создавать отчеты.

Отчет предназначен для создания документа на основе данных из таблицы или запроса. Этот документ можно распечатать или включить в документ другого приложения, например, Word или Excel.

С помощью MS Access можно создать отчеты, которые:
- Группируют данные,
- Выполняют различные групповые вычисления,
- Содержат подчиненные формы, подчиненные отчеты и диаграммы,
- Представляют данные в привлекательном виде с использованием рисунков, линий и специальных шрифтов.

Окно отчета может находиться в одном из трех режимов: режиме конструктора, режиме просмотра образца и режиме предварительного просмотра.

Режим конструктора предназначен для создания новых и изменения существующих отчетов.
Режим просмотра образца предназначен для предварительной оценки правильности шрифтового оформления и расположения элементов управления в отчете.
В режиме предварительного просмотра отчет выглядит на экране так, как он выглядел бы, будучи напечатан.

В отчете можно располагать различные элементы управления.

В MS Access существуют следующие типы элементов управления: поле, надпись, группа, переключатель, флажок, выключатель, поле со списком, список, кнопка, рисунок, присоединенная рамка объекта, свободная рамка объекта, набор вкладок, подчиненная форма/отчет, разрыв страницы, линия, прямоугольник и дополнительные элементы ActiveX.

Элементы управления могут быть связанными, свободными или вычисляемыми. **Связанный** элемент управления присоединен к полю базовой таблицы или запроса. Такие элементы управления используются для отображения, ввода или обновления значений из полей базы данных. Для **вычисляемого** элемента управления в качестве источника данных используется выражение. В выражении могут быть использованы данные из поля базовой таблицы или запроса для формы или отчета, а также данные другого элемента управления формы или отчета. Для **свободного** элемента управления источника данных не существует. Свободные элементы управления используются для вывода на экран данных, линий, прямоугольников и рисунков (в Help «**Общие сведения об элементах управления**»).

Если в отчете необходимо использовать элементы управления, отличные от «**Поле**», то их надо выбрать из «**Панели элементов**» (режим конструктора, меню «**Вид**» _ пункт «**Панель элементов**»).

Создание вычисляемых полей

1. Добавить элемент «**Поле**»,
2. В свойство поля «**Данные**» ввести знак равенства, а затем нужное выражение.

Например, **=Date()** в поле будет выведена текущая дата

=**Sum**([Стоимость]) суммирование данных по полю **Стоимость** из таблицы или запроса.

Обычно для сложных выражений используют мастер построения выражений.

3. Свойство «**Сумма с накоплением**» следует использовать для вычисления суммы, которая меняется при переходе к каждой следующей записи.

· **Для группы**. Значение сбрасывается до 0 в начале каждой группы более высокого уровня.

· **Для всего**. Сумма накапливается до конца отчета.

(в Help «**Вычисление в отчете суммы с накоплением**»).

Группировка записей в отчете

В отчетах допускается группировка по 10 и менее полям или выражениям (в Help «**Группировка записей в отчете**»).

1. Откройте отчет в режиме конструктора.
2. Меню «**Вид**» _ пункт «**Сортировка и группировка**», чтобы открыть окно «**Сортировка и группировка**».
3. Укажите порядок сортировки данных в отчете.
4. Выберите поле или выражение, для которого задаются параметры группировки.
5. Задайте значения свойств группы, перечисленные в следующем списке. Чтобы создать уровень группировки и определить другие свойства группы, необходимо задать значение «**Да**» по крайней мере, для одного из свойств «**Заголовок группы**» (GroupHeader) или «**Примечание группы**» (GroupFooter).

Заголовок группы (GroupHeader). Добавляет или удаляет заголовок группы, определяемой полем или выражением.

Примечание группы (GroupFooter). Добавляет или удаляет область примечаний группы, определяемой полем или выражением.

Группировка (GroupOn). Определяет способ группировки значений. Список доступных значений этого свойства зависит от типа данных поля, по которому проводится группировка. При группировке по выражению выводится полный список значений данного свойства.

Интервал (GroupInterval). Определяет любой допустимый интервал значений поля или выражения, по которому проводится группировка.

Не разрывать (KeepTogether). Задает или отменяет обязательную печать группы на одной странице.

Свойство «**Группировка**» для

- Текстового поля: «**По полному значению**» или «**По первым символам**»
- Поля даты/времени:

По полному значению По значению поля или выражения.
По годам По датам, принадлежащим к одному году.
По кварталам По датам, принадлежащим к одному кварталу.
По месяцам По датам, принадлежащим к одному месяцу.
По неделям По датам, принадлежащим к одной неделе.
По дням По одинаковым датам.
По часам По значениям времени с совпадающими часами.
По минутам По значениям времени с совпадающими часами и минутами.

Чтобы создать отчет следует в окне базы данных выбрать объект «**Отчеты**» и щелкнуть по кнопке «**Создать**». MS Access открывает окно «**Новый тчет**» и предлагает несколько способов создания отчета (рис.26).

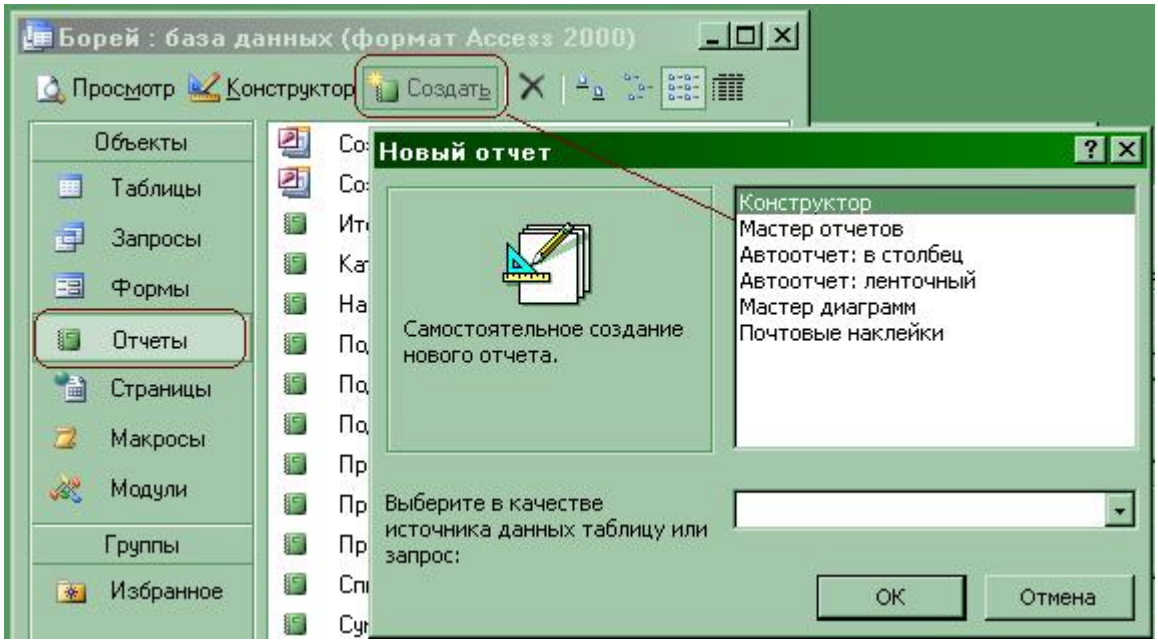


Рис. 26. Создание отчета

1. Создать отчет, показывающий список имеющихся в наличии товаров по алфавиту. Использовать таблицы **Типы** и **Товары**. В начале каждой группы печатать букву, с которой начинается наименование товара. В отчет включить поля **Марка, Категория, ЕдиницаИзмерения, НаСкладе**. Для вывода в отчет первой буквы товара использовать функцию

Left(поле_из_запроса/таблицы; количество символов)

В отчет включить 2 уровня группировки:

1. по полю **Марка**,
свойство «**Группировка**» установить в значение «**По первым знакам**»,
свойство «**Заголовок группы**» установить в значение «**Да**»,
свойство «**Примечание группы**» установить в значение «**Да**» (чтобы записи сгруппировать по буквам алфавита);

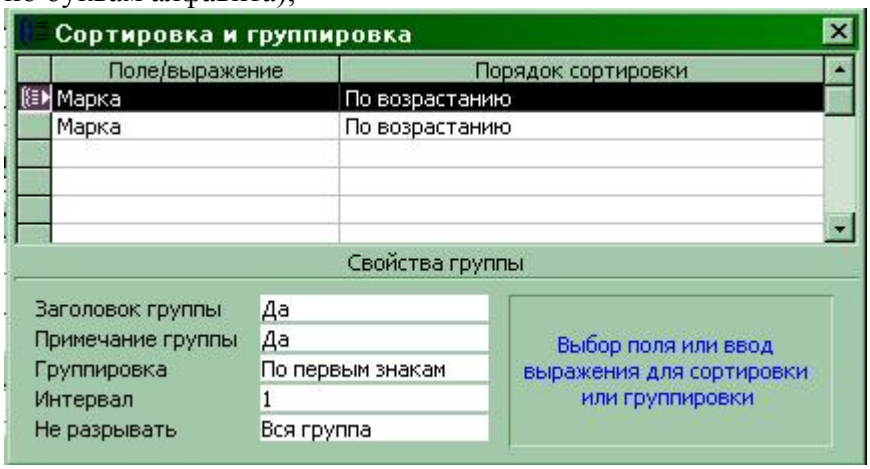


Рис. 27. Установка свойств группы

2. по полю **Марка**,
свойство «**Группировка**» установить в значение «**По полному значению**»,
свойство «**Заголовок группы**» установить в значение «**Нет**»,
свойство «**Примечание группы**» установить в значение «**Нет**» (чтобы записи отсортировать в пределах одной буквы алфавита).

(В базе Борей.mdb или NWIND.mdb отчет «**Список товаров**»).

Вычисления для одной записи и для группы записей

Предположим, что отчет построен по запросу, содержащему поля **НазваниеТовара**, **КодЗаказа**, **Цена**, **Количество**.

Для вычисления суммы за товар по каждой записи в область данных отчета поместить элемент «**Поле**» (вычислимый элемент управления). В свойство «**Данные**» для поля написать выражение **=[Цена]*[Количество**]

Для вычисления суммы по всем товарам группы **КодЗаказа**, в заголовок или примечание группы поместить элемент «**Поле**». В свойство «**Данные**» для поля написать выражение **=Sum([Цена]*[Количество**])

!!! В качестве аргумента выражения функции **Sum** можно использовать имена полей таблицы или запроса (в том числе вычисляемых полей запроса), **но не имена элементов управления**.

Для суммирования значений вычисляемых элементов управления необходимо в качестве аргумента функции **Sum** повторить выражение, которое использовалось при определении вычислимого элемента управления.

Например, в элементе управления =[Цена]*0.75
При подведении итога =Sum([Цена]*0.75)

При вычислении, чтобы гарантировать точность результатов, следует округлять полученные значения до 2-х знаков с помощью функций **CLng**, **Cint** или **Int.**

 2. Создать отчет о сумме продаж по годам с разбивкой по кварталам. Отчет должен содержать информацию о годе, квартале, количестве заказов за квартал, сумме заказов за квартал, количестве заказов за год, сумме заказов за год.
В качестве образца использовать в базе Борей.mdb или NWIND.mdb отчет «**Суммы продаж по годам**».

Объединение нескольких отчетов

Подчиненным отчетом называют отчет, вставленный в другой отчет. При комбинировании отчетов один из отчетов является главным. Главный отчет может быть как присоединенным, так и свободным, т.е. не базирующимся на таблице, запросе или инструкции SQL.

Свободный главный отчет может служить контейнером нескольких не связанных между собой отчетов, которые требуется объединить.

Главный отчет связывают с таблицей, запросом или инструкцией SQL в тех случаях, когда в него требуется вставить подчиненные отчеты, в которых выводятся данные связанные с данными в главном отчете. Например, в главном отчете могут быть выведены все записи о продажах за год, а в подчиненном отчете итоговые данные, например, суммы продаж за каждый квартал.

В главном отчете могут также содержаться данные, являющиеся общими для двух или нескольких подчиненных отчетов. В этом случае области данных выводятся в подчиненных отчетах.

В главный отчет наряду с подчиненными отчетами включают также подчиненные формы, причем число таких подчиненных форм не ограничивается. Более того, главный отчет может содержать подчиненные формы или отчеты двух уровней вложенности. Например, в отчете может содержаться подчиненный отчет, который в свою очередь содержит подчиненную форму или подчиненный отчет. Возможные комбинации

подчиненных форм и отчетов демонстрируются в следующей таблице (в Help «**Общие сведения о подчиненных отчетах**»).

Этапы создания отчета:

1. Создать главный отчет (объект «**Отчеты**» и щелкнуть по кнопке «**Создать**»).
2. Создать подчиненные отчеты обычным образом (объект «**Отчеты**» и щелкнуть по кнопке «**Создать**»).
3. Вставка подчиненного отчета в главный.

3.1. Главный отчет должен находиться в режиме конструктора.

3.2. Добавить в соответствующий раздел элемент управления «**Подчиненная форма/ подчиненный отчет**».

3.3.Подчиненному отчету указать свойства «**Объект – источник**», «**Основные поля**» и «**Подчиненные поля**».

Если в подчиненном отчете требуется связь с главным отчетом, в бланке свойств подчиненного отчета определить свойства «**Основные поля**» и «**Подчиненные поля**» (рис. 28).

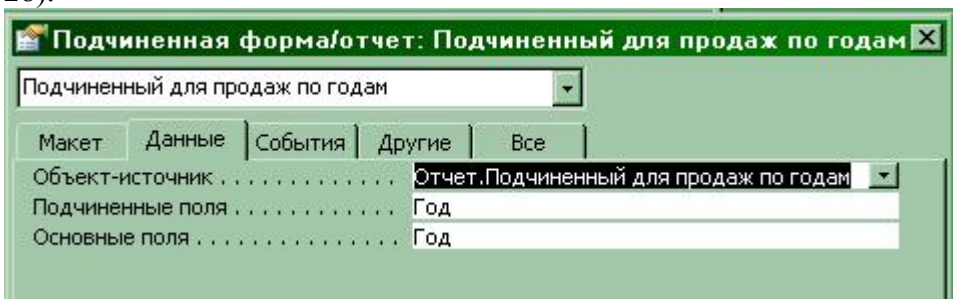


Рис. 28. Определение свойств для элемента «Подчиненная форма/ подчиненный отчет».

3. Создать отчет аналогичный отчету из базы Борей или NWIND «**Продажи по годам**».

Главный отчет отражает информацию по каждому заказу, исполненному в течение года. Подчиненный отчет содержит информацию о том сколько заказов исполнено и на какую сумму по году в разбивке по кварталам.

Обратите внимание на то, что форма с вводом периода дат вызывается из события Открытие для отчета. А значения введенных дат являются параметрами запроса «**Продажи по годам**». В свою очередь запрос «**Продажи по годам**» является источником для главного отчета. Главный и подчиненный отчеты связаны по году (поле **Год**).

Вывод данных в несколько столбцов

Если данные в отчете необходимо выводить в несколько столбцов, то установите Параметры страницы (меню «**Файл**» пункт «**Параметры страницы**», вкладка «**Столбцы**», задать значения для число столбцов, интервал строк, столбцов.

4. Вывести список товаров по маркам в 2 столбца.

Образцом является отчет из базы Борей «**Товары по типам**».

Лабораторная работа №7

Работа с отчетами

Цель: Приобрести умения и навыки при работе с отчетами. Научиться создавать отчеты и запускать их из формы.

 1. Создать запрос **ЗаказыКлиентов** из таблиц **Заказы** (взять поля **КодЗаказа**, **ДатаРазмещения**), **Типы** (поле **Категория**), **Товары** (поле **Марка**), **Заказано** (поля **Цена**, **Количество**, **скидка**), **Клиенты** (поле **Название**).

Создание отчета с помощью конструктора

 2. Создать отчет **ЗаказыКлиентов**, в котором информация должна быть представлена по заказам с позициями товаров в заказе. Вычислять через отчет сумму по каждому товару и по каждому заказу.

2.1. Объект «**Отчеты**» и щелкнуть по кнопке «**Создать**». Выбрать «**Конструктор**». В качестве источника выбрать запрос **ЗаказыКлиентов**.

2.2. В «**Область данных**» добавить необходимые поля через элемент «**Поле**» с панели элементов или через пункт «**Список полей**» в меню «**Вид**». Добавьте поле для суммы по товару.

2.3.В «**Верхний колонтитул**» добавьте надпись отчета через элемент «**Надпись**» с панели элементов. Просмотрите отчет в режиме просмотра.

2.4. Меню «**Вид**» _ пункт «**Сортировка и группировка**». Выбрать поле **КодЗаказа**. В «**Свойствах группы**» выбрать «**Заголовок группы**» и установить его значение в «**Да**», «**Примечание группы**» установить его значение в «Да».

2.5. Из «**Области данных**» в «**Заголовок группы КодЗаказа**» перенести поля **КодЗаказа**, **Название** (клиента), **ДатаРазмещения**. Просмотреть отчет в режиме просмотра.

2.6. В поле **Сумма** вычислить значение итоговой суммы по товару.

2.7. В «**Примечание группы КодЗаказа**» добавить поле для итоговой суммы по заказу. Использовать выражение, вычисляющее сумму по каждому товару. В итоговую сумму написать **=Sum(...выражение..).** Просмотреть отчет.

2.8. В «**Верхний колонтитул**» добавить надписи для полей (элемент «**Надпись**» с панели элементов). Организовать красивое форматирование отчета (Свойства отчета, вкладка «**Макет**»).

2.9. Сделать дополнительную группировку по полю **ДатаРазмещения**, добавив заголовок и примечание группы. В «**Примечание группы**» добавить поле для суммы по **ДатаРазмещения**. В окне «**Сортировка и группировка**» поле **ДатаРазмещения** перенести выше поля **КодЗаказа**. Просмотреть отчет.

Открытие отчета

Метод **OpenReport** выполняет макрокоманду «**ОткрытьОтчет**» (OpenReport) в программе Visual Basic.
Синтаксис

DoCmd.OpenReport имяОтчета [, режим] [, имяФайла] [, условиеWhere]

Параметры (аргументы) в квадратных скобках являются необязательными. Метод OpenReport использует следующие аргументы.

Аргумент	Описание
имяОтчета	Строковое выражение, представляющее допустимое имя отчета в текущей базе данных.
режим	Одна из следующих встроенных констант: acViewDesign (конструктор) acViewNormal (печать, значение по умолчанию) acViewPreview (просмотр) Константа acViewNormal задает немедленный вывод отчета на печать.

	Если оставить данный аргумент пустым, подразумевается значение по умолчанию (acViewNormal).
имяФайла	Строковое выражение, представляющее допустимое имя запроса в текущей базе данных.
условиеWhere	Строковое выражение, представляющее допустимое предложение SQL WHERE без ключевого слова WHERE.

Максимальная длина строки в аргументе **условиеWhere** составляет 32768 символов (в отличие от аргумента «**Условие отбора**» в окне макроса, максимальная длина которого составляет 256 символов).

Необязательный аргумент посреди списка аргументов разрешается пропустить, однако, при этом необходимо ввести запятую, представляющую пропущенный аргумент. Если опускаются один или несколько последних аргументов, вводить запятые вслед за последним указанным аргументом не требуется.

В следующем примере отчет «Продажи» печатается с использованием запроса «Фильтр для отчета»:

DoCmd.OpenReport "Продажи", acViewNormal, "Фильтр для отчета"

 3. Создать форму **ЗапускОтчета,** в которой разместить два поля дат (**ДатаНачала** и **ДатаКонца**) и кнопку, по которой будем вызывать отчет. Использовать команду **Docmd.OpenReport**, поля **ДатаНачала** и **ДатаКонца** включить в строку условия команды.
«**Создать**»
3.1. Добавить поля для границ дат в верхний колонтитул отчета. Значения в эти поля класть из полей формы или работать через глобальную переменную (Создать модуль (Объект «**Модули**» и щелкнуть по кнопке «**Создать**»). В модуле создать глобальную переменную и написать две функции: положить в глобальную переменную и взять из глобальной переменной. Использовать эти функции в форме и отчете). Для дат необходимо делать преобразование дат через функцию **Format** (смотри файл справки).
3.2. В форму **ЗапускОтчета** добавить поле со списком для выбора клиента. Использовать это поле в строке условия для отчета.
3.3. Если работает мастер отчетов, то создать отчет аналогичный предыдущему.

Лабораторная работа №8

Элементы управления на форме

Цель: Приобрести умения и навыки при работе с элементы управления на форме. Научиться создавать элементы управления и работать с ними из VBA.

1. Поле со списком или список

Чтобы указать какие строки должны быть включены в список или поле со списком, следует использовать свойства «**Тип источника строк**» (RowSourceType) и «**Источник строк**» (RowSource).

Чтобы включить в список	Значение свойства «**Тип источника строк**»	Значение свойства «**Источник строк**»
Строки из таблицы или запроса	Таблица/запрос (по умолчанию)	Имя таблицы или запроса
Строки, полученные с помощью инструкции SELECT SQL	Таблица/запрос (по умолчанию)	инструкция SQL

Список значений, указанных пользователем	Список значений	Список значений, разделенных точками с запятой
Имена полей таблицы или запроса	Список полей	Имя таблицы или запроса
Значения, полученные с помощью определяемой пользователем функцию VBA	Имя функции	(пусто)

Можно эти свойства задать программно.
В следующем примере для свойства «**Тип источника строк**» (RowSourceType) поля со списком задается «**Таблица/запрос**», а в свойстве «**Источник строк**» (RowSource) указывается запрос «СписокСотрудников».

Forms!Форма1!ПолеСоСписком2.RowSourceType = "Таблица/запрос"
или
Forms!Форма1!ПолеСоСписком2.RowSourceType ="Table/Query"

в зависимости от версии MS Access (Использовать клавишу F1 на RowSourceType в окне Visual Basic).

Forms!Форма1!ПолеСоСписком2.RowSource = "СписокСотрудников"

Свойства элемента управления «**Поле со списком**» или «**Список**»

Имя элемента управления	Вкладка «**Другие**»_ «**Имя**»	Имя элемента на форме
Данные	Вкладка «**Данные**»_ «**Данные**»	Это поле из таблицы или запроса, которое указано в качестве источника строк формы, в которой находится этот элемент управления.
Число столбцов	Вкладка «**Все**»_ «**Число столбцов**»	Количество столбцов в списке (н-р, 3)
Ширина столбцов	Вкладка «**Все**»_ «**Ширина столбцов**»	Ширина ст.1; ширина ст.2;…; 0 означает, что столбец не будет виден в списке
Заглавия столбцов	Вкладка «**Все**»_ «**Заглавия столбцов**»	Да или нет
Присоединенный столбец	Вкладка «**Данные**»_ «**Присоединенный столбец**»	Значение этого столбца попадет в таблицу или запрос в поле, указанное в свойстве «**Данные**»
Ограничиться списком (имеет смысл, если связанный столбец один и он изображается в поле со списком)	Вкладка «**Данные**»_ «**Ограничиться списком**»	«**нет**» тогда в это поле можно вводить значения

Свойство «**Column**» определяет конкретный столбец или комбинацию столбца и строки элемента управления (поля со списком или списка). Синтаксис свойства «**Column**»:

Control.Column(column, row)

Элемент	Описание
Control	Обязательный элемент. Объект, представляющий собой активный список или поле со списком
Column	Обязательный элемент. Целое число в диапазоне от 0 до значения свойства ColumnCount минус 1, указывающее столбец (нумерация столбцов с нуля).
Row	Необязательный элемент. Целое число в диапазоне от 0 до значения свойства ListCount минус 1, указывающее строку (нумерация строк с нуля).

Это свойство может использоваться, например, для присвоения значения указанного элемента списка значению поля или другого элемента управления.

Если Вы создали элемент управления «**Список**», и необходимо из списка выбирать сразу несколько значений, то в свойстве «**Несвязанное выделение**» следует выбрать «**Со связанным выбором**» (рис. 29).

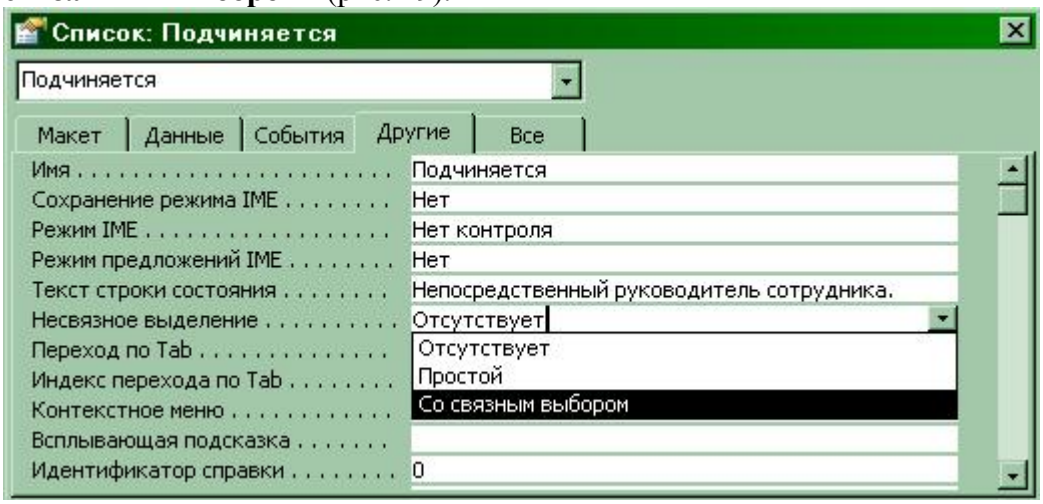
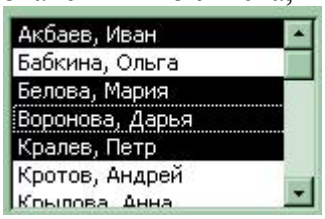


Рис. 29. Установка свойства «**Несвязанное выделение**»

Теперь удерживая нажатой клавишу CTRL и щелкая левой кнопкой мыши по значениям из списка, Вы сможете выбрать сразу несколько значений.



Все выбранные элементы попадут в коллекцию «**ItemsSelected**». Доступ к этой коллекции осуществляется через свойство «**ItemsSelected**», возвращающее ссылку на семейство «**ItemsSelected**», которое содержит в отличие от других семейств не объекты, а значения типа Variant. Эти значения представляют собой целочисленные индексы, указывающие положение выделенной строки в списке.

Свойство «**ItemData**» возвращает значение, содержащееся в присоединенном столбце указанной строки элемента управления «**Список**» (ListBox). Синтаксис свойства «**ItemData**»:

control.ItemData(rowindex)

Элемент	Описание
Control	Обязательный элемент. Объект, представляющий собой Список (ListBox) или Поле со списком (ComboBox)

Rowindex	Обязательный элемент. Целое число в диапазоне от 0 до значения свойства ListCount минус 1, определяющего строку, из которой вы хотите получить значение

Представленные свойства можно использовать для получения данных из выделенных строк списка.

```
Sub RowsSelected()
   Dim ctlList As Control, varItem As Variant

   ' создать объект Control для списка
   Set ctlList = Forms!Employees!EmployeeList
   ' Пройти по выбранным элементам списка
   For Each varItem in ctlList.ItemsSelected
      ' Вывести значения
      Debug.Print ctlList.ItemData(varItem)
   Next varItem
End Sub
```

 1. Создать элемент управления «**Поле со списком**» и программно получить и использовать его значение.

 2. Создать элемент управления «**Список**» с возможностью выбора сразу нескольких значений. Программно получить и как-то отобразить выбранные значения.

2. Флажки, переключатели, выключатели

В установленном состоянии эти элементы соответствуют значению «**Да**» или «**истина**», а в снятом – «**Нет**» или «**ложь**».
Пример использования флажка **Сведения** в коде программы:

```
If Forms![Продажи по годам]!Сведения Then
      Me!Отображение.Value = False
   Else
      Cancel = True
   End If
```

 3. Создать элементы управления «**Флажок**», «**Переключатель**», «**Выключатель**» и программно получить и использовать их значения.

3. Группа

Группа используется для выбора из ограниченного набора. Группа содержит несколько флажков или переключателей или выключателей. Каждый элемент в группе имеет уникальное числовое значение (свойство «**Значение параметра**»). После выбора элемента из группы, значение группы станет равным значению выбранного элемента.

Пример

В базе Борей.mdb (nwind.mdb) в форме **Телефоны клиентов** в элементе **группа ОтборКлиентов** после обновления значения группы запускается макрос **Телефоны клиентов**.

Рис. 30. Пример группы выключателей

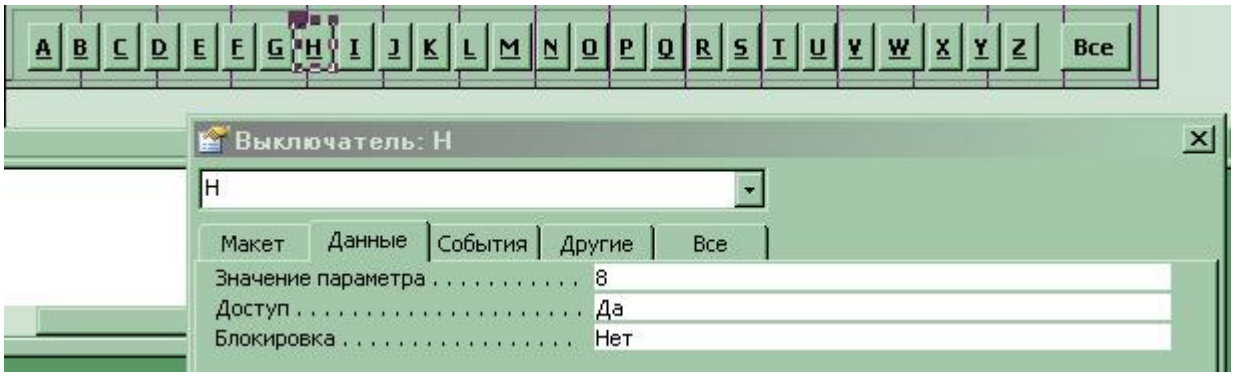


Рис. 31. Выключатель в группе

Еще пример работы со значением элемента управления «**Группа**»
Select Case Me!ОтборКлиентов
Case 1
MsgBox "A"
Case 2
MsgBox "B"
…
End Select

 4. Создать элемент управления «**Группа**» и программно получить и использовать его значение.

Лабораторная работа №9

Модель доступа к данным ADO

Цель: Приобрести умения и навыки доступа к данным, используя модель ADO.

Модель доступа к данным **ADO** (ActiveX Data Objects) является средством доступа к данным различного типа, появившимся первоначально в Microsoft Access 2000. Модель ADO опирается на услуги так называемого универсального поставщика данных OLEDB (OLE DB provider) и использует новые технологии программирования. OLE DB позволяет получать унифицированный доступ как к реляционным источникам данных, так и к нереляционным источникам, таким, как хранилища различной неупорядоченной информации (почта, различные текстовые наборы, графические данные, службы каталогов и т. д.). OLE DB позволяет использовать одну и ту же модель доступа к данным (ADO) для связи с

практически всеми известными источниками данных (в том числе и с хранилищами (нереляционными источниками данных)).

Для использования ADO Вам следует подключить ссылку на соответствующую библиотеку. Для этого нужно в окне редактора Visual Basic установить ссылку на библиотеку Microsoft ActiveX Data Objects (меню «**Tools**» _ «**References**»).

Откроется диалоговое окно «**References …**» (Ссылки), в котором следует выбрать библиотеку «**Microsoft ActiveX Data Objects 2.8**» (или старше). Соответствующий файл библиотеки называется **msado28.tlb**. Эти файлы обычно расположены в каталоге Programs Files/Common Files/System/Ado.

В объектную модель **ADO** входят следующие объекты:
Connection – открывает сеанс обмена данными,
Command- представляет собой инструкцию SQL,
Parameter - представляет собой параметр инструкцию SQL,
Parameters – содержит все объекты Parameter, ассоциированные с объектом Command,
Recordset – представляет собой набор записей и позволяет осуществлять навигацию по записям и манипулировать с данными в нем,
Field- представляет собой поле (столбец) в наборе записей Recordset,
Fields - содержит все объекты Field, ассоциированные с объектом Recordset,
Error - представляет собой информацию об ошибке, произошедшей во время сеанса связи,
Errors – все объекты Error в этом семействе создаются в ответ на одну ошибку, произошедшую во время сеанса связи,
Property – представляет характеристику (свойство) любого объекта ADO,
Properties – содержит все объекты Property, ассоциированные с объектами **Connection,** Command, Recordset или Field.

Основными объектами в **ADO** являются **Connection, Command, Recordset**.

Объект Connection

Объект **Connection** отвечает за физический сеанс связи с конкретным источником данных. С помощью объекта **Connection** следует конфигурировать сеанс связи до использования данных подключаемого источника данных; указать базу данных, которая будет использоваться по умолчанию в качестве источника данных; указать провайдера OLE DB, который будет использоваться для связи с данными; установить уровень изоляции для транзакций; реально устанавливать и прерывать связь с указанным источником данных. Кроме того можно выполнить достаточно многих других действий, связанных с различными аспектами использования необходимого источника данных.

Рассмотрим некоторые свойства и методы объекта **Connection.**

Свойство **ConnectionString** (строковое значение). Одно из наиболее важных и постоянно используемых свойств. Свойство имеет статус "чтение/запись". Информация, заданная в тексте строки, используется для установления соединения с источником данных. Синтаксически строка соединения представляет пары вида: **аргумент = значение**, разделенные символом ";" (точка с запятой). **ADO** поддерживает пять аргументов, но в зависимости от Провайдера ему могут предаваться и другие аргументы, которые никак не обрабатываются средствами **ADO** и передаются непосредственно Провайдеру. Вот список общих для всех Провайдеров аргументов, поддерживаемых **ADO**:
Provider - имя Провайдера, с которым устанавливается соединение.
File Name - имя файла, содержащего предустановленную информацию о соединении, передаваемое провайдеру.
Remote Provider - имя Провайдера, используемое при открытии соединения на клиентской стороне. Используется только при работе со службой RDS.
Remote Server - имя сервера (путь), используемое при открытии соединения на клиентской стороне. Используется только при работе со службой RDS.

URL - адрес, идентифицирующий такие ресурсы, как файл или каталог. Заметьте, свойства, установленные в строке соединения, могут измениться после открытия соединения, поскольку может произойти трансляция аргументов в форму, предусмотренную Провайдером.

Ниже приведено несколько примеров различных строк подключения (**Connection String**).

Использование SQL Server OLE DB Provider (подключение Windows к SQL Server):

"Provider=SQLOLEDB;Data Source=YourDb;Initial Catalog=pubs"

"Provider=SQLOLEDB;Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=Northwind;Data Source=london1"

"Provider=SQLOLEDB;Data Source=MyServer;Initial Catalog=MyBase;UserID=ItsMe;Password=MyPass;"

Строка подключения к источнику данных ODBC

"Provider=MSDASQL;DSN=MyDSN; UID=ItsMe;PWD=MyPass; "

При подключении к файлу Access или Excel строка подключения могла бы выглядеть так:

"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=D:\Борей.mdb"

Методы объекта **Connection** позволяют открывать или закрывать соединение, выполнять команды, основанные на SQL-операторах и т.д. Далее следует краткое описание некоторых методов.

Метод **Open** устанавливает физическое соединение с источником данных:

connection.Open ConnectionString, UserID, Password, Options

В этом методе можно указать следующие (необязательные параметры):
ConnectionString – Строка подключения (смотри свойство **ConnectionString**);
UserID – имя пользователя, который устанавливает соединение;
Password – пароль пользователя, который устанавливает соединение

После создания экземпляра объекта **ADODB.Connection** используем следующие команды для открытия базы данных:

```
'команда открывает базу данных Microsoft Access 2003
objConnection.Open  "Provider = Microsoft.Jet.OLEDB.4.0; Data Source = C:\Scripts\Inventory.mdb"

'команда открытия базы данных Access 2007
objConnection.Open   "Provider = Microsoft.ACE.OLEDB.12.0; Data Source = C:\Scripts\Inventory.accdb"

'для открытия базы данных SQL Server
objConnection.Open  "Provider=SQLOLEDB;Data Source=atl-sql-01;Trusted_Connection=Yes;Initial Catalog=Inventory;User ID=fabrikam\kenmyer;Password=34DE6t4G!;"
```

Методы **BeginTrans, CommitTrans и RollbackTrans**. Эти методы нужны, если вы в своих приложениях используете механизм транзакций. В одну транзакцию объединяется

группа действий, которые либо должны быть целиком выполнены, либо целиком отменены. В такую группу, например, можно объединить операцию снятия денег со счета и операцию помещения их на другой счет. Если выполнять эти операции разрозненно, то в случае сбоя может случиться ситуация, при которой деньги будут сняты с одного счета, а на другой счет не поступят (то есть просто исчезнут из учета). Применяя транзакции, в такой ситуации можно сделать откат (отмену всей группы операций), в результате чего будет отменено и снятие денег со счета (первая операция).

При помощи метода **BeginTrans** вы можете начать транзакцию, а при помощи метода **CommitTrans** — подтвердить все сделанные транзакцией изменения. Если в процессе выполнения операций, включенных в транзакцию, пошло что-то не так, откат (отмену всех изменений) можно сделать при помощи метода **RollbackTrans.** Метод **BeginTrans** можно вызывать как процедуру и как функцию. Во втором случае **BeginTrans** возвращает значение, соответствующее текущему уровню изоляции транзакции (см. соответствующее свойство, описанное чуть выше):

Object.BeginTrans

Level= Object.BeginTrans

Метод **Cancel** объекта **Connection** прерывает выполнение асинхронно запущенных методов **Open** или **Execute**, в зависимости от того, какой из них был запущен последним. Асинхронно запущенные методы — это методы, которые были запущены с асинхронными опциями (то есть с опциями **adAsyncConnect**, **adAsyncExecute, adAsyncFetch**).

Метод **Close** закрывает объект **Connection** и все связанные с ним объекты, освобождая занятые ими системные ресурсы. Чтобы окончательно удалить объект из памяти, как и в случае использования других объектных переменных, следует присвоить переменной значение Nothing:

MyConn = Nothing.

При помощи метода **Execute** можно выполнить запрос, SQL-оператор, хранимую процедуру или любую команду, распознаваемую конкретным провайдером данных. Формат запуска этого метода имеет два варианта.

Set recordset = connection.Execute (CommandText, RecordsAffected, Options)

connection.Execute CommandText, RecordsAffected, Options

Первый вариант соответствует команде, возвращающей набор записей. Соответственно в этом случае метод запускается как функция, возвращающая ссылку на объект типа **Recordset**. Второй вариант, соответствующий формату запуска процедуры, применяется для запуска команд, не возвращающих записи (например, для запуска выполняемых запросов или соответствующих хранимых процедур). В любом случае метод принимает три аргумента.

CommandText аргумент типа String может содержать SQL-оператор, имя таблицы или сохраненной процедуры, а также текст ориентированной на специфику конкретного провайдера данных команды, которую необходимо выполнить.

RecordsAffected аргумент является необязательным. Здесь вы можете указать переменную типа Long, которая после выполнения команды будет содержать количество записей, подвергшихся воздействию команды (возвращаемых командой, измененных командой и т.д.).

Options аргумент Options, также необязательный, указывает на способ трактовки текста команды (первый аргумент). Может содержать одну или комбинацию нескольких констант типа **CommandTypeEnum** или **ExecuteOptionEnum**. Констант этих типов достаточно много. С полным перечнем вы сможете ознакомиться при помощи справочной системы MS Access. В качестве примера можно упомянуть следующие: **adCmdTableDirect** (первый аргумент трактуется как имя таблицы, чьи строки необходимо возвратить), **adCmdStoredProc** (первый

аргумент метода содержит имя хранимой на стороне сервера процедуры), **adAsyncExecute** (указывает на то, что команда должна выполняться асинхронно, то есть приложение не будет ожидать окончания выполнения команды) и т.д. Ниже приведен простой пример применения метода **Execute.**

```
Dim MyConn As ADODB.Connection
Dim MyStr As String
Dim ConStr As String

MyStr ="Delete * from [Временная];"
ConStr="Provider = Microsoft.Jet.OLEDB.4.0; Data Source = C:\db2.mdb"
Set MyConn= New ADODB. Connection
MyConn.Open ConStr
MyConn.Execute MyStr
MyConn.Close
```

В данном примере осуществляется подключение к базе данных MS Access db2.mdb, расположенной в корневом каталоге диска C. Обратите внимание на то, что перед подключением к базе данных вы должны указать как минимум имя провайдера и название файла базы данных, иначе вы получите сообщение об ошибке.

Метод **OpenSchema** возвращает ссылку на объект **Recordset**, представляющий определенную схему.

Set recordset = connection.OpenSchema (QueryType, Criteria, SchemaID)

Тип этой схемы определяется первым аргументом метода, который может принимать одно из значений типа **SchemsEnum**. К этому типу принадлежит достаточно большое количество именованных констант. В данной ситуации схема имеет достаточно широкое толкование. Например, указав константу **adSchemaTables,** вы получаете список доступных вам таблиц. Список констант действительно очень большой, и если вы действительно интересуетесь этим вопросом, вам следует воспользоваться справкой.

Объект CurrentProject

Для работы с текущей базой данных можно использовать объект **CurrentProject.**

Его метод **OpenConnection** позволяет открыть ADO соединение к текущей базе данных.

CurrentProject.OpenConnection(BaseConnectionString, UserID, Password)

Все параметры являются необязательными.

BaseConnectionString - строковое выражение, которое определяет строку соединения с базой данных.

Свойство **Connection** возвращает ссылку на текущий объект **Connection.**

```
Dim MyConn As ADODB.Connection
MyConn= CurrentProject. Connection
```

Объект Recordset

Объект **Recordset** представляет любой набор записей. Этот набор записей можно представлять себе как обычную временную таблицу, имеющую строки и столбцы. При помощи этого объекта вы сможете манипулировать данными, полученными от установленного провайдера. Каждый объект **Recordset** характеризуется определенным

типом курсора, то есть типом объекта, обслуживающего данный набор. В **ADO** определено четыре типа курсора, один из которых указывается при открытии объекта **Recordset:**

Dynamic cursor (динамический курсор). Позволяет просматривать все изменения данных (вставку, удаление, редактирование записей), сделанные другими пользователями. Существует возможность перемещаться в любом направлении по набору, а также пользоваться закладками (если их поддерживает провайдер).

Keyset cursor (курсор типа ключевой набор). Отличается от динамического курсора тем, что для просмотра недоступны добавленные или удаленные другими пользователями записи.

Static cursor (статический курсор). Вам предоставляется статическая копия набора данных, которую вы можете использовать только для просмотра, генерации отчетов и тому подобных действий.

Forward-only cursor (курсор с возможностью перемещения только вперед). Курсор этого типа позволяет перемещаться по набору данных только вперед. Все изменения в наборе, сделанные другими пользователями, вам будут невидны. Курсоры этого типа применяются для оптимизации работы приложения в тех случаях, когда требуется только однократный проход по набору данных.

Для создания объекта **Recordset**используется следующий синтаксис:

```
Dim MyRec As ADODB.Recordset
Set MyRec = New ADODB.Recordset
```

С конкретным соединением (объектом **Connection**) и источником записей (таблицей, запросом и т.д.) объект **Recordset** связывается при его открытии (метод **Open**).

Рассмотрим некоторые свойства объекта **Recordset.**

Свойство **ActiveConnection** указывает на соединение (объект **Connection**), которому принадлежит открытый объект **Recordset**. Свойство доступно не только для чтения, но и для записи, то есть с его помощью можно связать объект **Recordset** с объектом **Connection**. Если объект **Connection** открыт, то в качестве значения свойства указывается значение типа Variant, содержащее его имя. Если соединение закрыто (не открыто), то значением свойства **ActiveConnection** должна быть строка, определяющая соединение (см. свойство **ConnectionString** объекта **Connection**).

Свойства **BOF** и **EOF.** Свойство **BOF** получает значение **True (Истина**), если была сделана попытка перейти на позицию, предшествующую первой записи (например, при итерации от конца набора данных в направлении его начала). Соответственно, свойство **EOF** получает значение **True (Истина)**, если была сделана попытка перейти на запись, следующую за последней записью набора. Если набор данных пуст, оба свойства имеют значение **True (Истина**). Свойства **BOF** и **EOF** часто используются для организации перебора записей набора (итерации по записям).

Свойство **CursorType** задает один из четырех типов курсора, который будет использоваться с набором данных (объектом **Recordset**). Выше уже перечислялись определенные в **ADO** типы курсора: **adOpenDynamic** (динамический набор), **adOpenKeyset** (ключевой набор), **adOpenStatic** (статический набор) и **adOpenForwardOnly** (набор с итерацией только вперед). Наиболее универсальным и часто употребляемым является курсор типа динамический (**adOpenDynamic**).

Свойство **LockType** указывает на тип блокировки, применяемый к редактируемым записям при одновременном доступе к ним нескольких пользователей. Значением этого свойства может быть одна из констант типа **LockTypeEnum**, к числу которых принадлежат следующие: **adLockBatchOptimistic** (оптимистический тип блокировки записей с отложенным обновлением), **adLockOptimistic** (оптимистический тип блокировки; записи блокируются только тогда, когда вызывается метод **Update**), **adLockPessimistic** (пессимистический тип блокировки; записи блокируются сразу после начала редактирования), **adLockReadOnly** (записи предназначены только для чтения; вы не можете изменять данные, хранящиеся в записях), **adLockUnspecified** (тип блокировки не определен;

провайдер данных использует тип блокировки, принятый для него по умолчанию). Вы должны установить это свойство до открытия набора данных (объекта **Recordset**). Кроме того, вы должны помнить о том, что не все провайдеры данных поддерживают все типы блокировок, определенных в **ADO**.

Свойство **Source** указывает на источник данных объекта **Recordset**. Если объект **Recordset** открыт, то свойство **Source** доступно только для чтения. В качестве значения свойства можно указать либо строку, содержащую имя базовой таблицы, запроса, хранимой процедуры или SQL-оператор, либо имя объекта **Command**. Возвращает свойство **Source** строковое значение, указывающее на источник данных.

Свойство **State** только для чтения, возвращающее значение, которое описывает текущее состояние объекта **Recordset**. Для этого объекта может быть возвращено одно из трех значений (констант) типа **ObjectStateEnum**: **adStateClosed** (набор данных закрыт), **adStateOpen** (набор данных открыт), **adStateFetching** (набор данных получает записи).

Методы объекта **Recordset** позволяют манипулировать как отдельными записями набора данных, так и целиком всем набором. С их помощью вы можете организовать поиск необходимых данных и перемещение по набору данных, а также производить много других действий.

Прежде чем производить какие-либо действия с набором данных (объектом **Recordset**), его необходимо сначала открыть при помощи метода **Open**.

Метод **Open** служит для открытия набора данных (объекта **Recordset**). После его успешного применения вы сможете модифицировать записи набора, перемещаться по его записям, осуществлять поиск и т.д. Формат запуска метода Open таков:

MyRec.Open Source, ActiveConnection, Cursor-Type, LockType, Options

MyRec — объявленная ранее переменная типа **Recordset**. Метод **Open** принимает пять аргументов. Все пять аргументов являются необязательными, однако первые два должны быть обязательно указаны ранее (например, при помощи свойств **Source** и **ActiveConnection**). Если аргумент пропущен и правее его есть еще хотя бы один аргумент, то ограничивающую запятую все равно нужно указывать. Например,

MyRec.Open "Временная", , adOpenDynamic

Этот пример будет работать только в том случае, если ранее вы задали открытое активное соединение (объект **Connection**) при помощи свойства **ActiveConnection** объекта **Recordset**. Первый аргумент метода **Open** кроме имени таблицы активного подключения, может содержать также имя запроса, хранимой процедуры, объекта **Command**, строку SQL-оператора, имя объекта **Stream** или файла, содержащего ранее сохраненный объект Recordset, а также адрес URL. Второй аргумент может содержать либо имя подходящего объекта **Connection**, либо строку подключения (см. свойство **ConnectionString** объекта **Connection**). При помощи третьего аргумента **CursorType** можно указать тип открываемого набора данных: **adOpenDynamic**, **adOpenStatic**, **adOpenKeyset** или **adOpenForwardOnly**. Аргумент **LockType** может содержать одну из констант типа **LockTypeEnum**. Аргумент **Options** может содержать одну или несколько констант типа **CommandTypeEnum** (например, **adCmdTable** (имя таблицы), **adCmdStoredProc** (имя хранимой процедуры)) или **ExecuteTypeEnum** (например, **adAsyncExecute** (первый аргумент задает асинхронно выполняемую команда), **adExecuteNoRecords** (в первом аргументе задано имя команды или хранимой процедуры, не возвращающей записи)).

Рассмотрим пример открытия набора записей. В этом примере не выполняются никакие действия с набором данных, набор просто сначала открывается, а затем закрывается. Все остальные манипуляции с набором данных должны проводиться между вызовами методов **Open** и **Close** объекта **Recordset**.

```
Dim MyConn As ADODB.Connection
Dim MyRec As ADODB.Recordset
Dim ConStr As String
```

```
ConStr="Provider = Microsoft.Jet.OLEDB.4.0; Data Source = C:\db2.mdb"
Set MyConn= New ADODB. Connection
Set MyRec= New ADODB.Recordset

MyConn.Open ConStr
MyRec.Open "Временная", MyConn, adOpenDynamic, adLockOptimistic
MyRec. Close
MyConn.Close
```

Если не указать тип блокировки явно, то по умолчанию будет принят тип блокировки **adLockReadOnly**. В результате вы не сможете добавлять, удалять или другим образом модифицировать записи.

Метод **AddNew** позволяет добавить запись в обновляемый набор данных.

recordset.AddNew Fields, Values

Метод может принимать два необязательных аргумента: **Fields** и **Values**. Аргумент **Fields** может быть именем поля, массивом полей или индексом полей базовой таблицы или запроса. Аргумент **Values** содержит значения для полей вставляемой записи и должен выглядеть в соответствии с первым аргументом. Если аргумент **Fields** содержит массив имен полей, то и аргумент **Values** должен быть массивом того же размера. Впрочем, такой синтаксис применяется редко. Чаще используется синтаксис, используемый в приведенном ниже примере.

```
MyRec.AddNew
MyRec![КодКлиента]=”SS55F”
MyRec![СтоимостьЗаказа]=567.89
MyRec.Update
```

Обратите внимание на то, что каждому методу **AddNew** должен соответствовать свой метод **Update** или **UpdateBatch** (если вы используете механизм отложенных обновлений).

Вызов метода **CancelUpdate** отменяет все изменения, сделанные в текущей или новой записи набора данных (объекта **Recordset**). Однако вызывать этот метод нужно до вызова соответствующего метода **Update**.

Вызов метода **Close** закрывает объект **Recordset**, после чего никаких действий с набором данных, кроме повторного открытия, производить нельзя.

Метод **Delete** удаляет записи из набора данных.

recordset.Delete AffectRecords

Метод **Delete** может принимать один аргумент типа **AffectEnum**. К этому типу относится четыре константы, но с методом **Delete** можно использовать только две из них: **adAffectCurrent** (удаляется только текущая запись; значение по умолчанию), **adAffectGroup** (удаляются записи, возвращаемые **Установленным фильтром**; фильтр должен быть установлен с одной из констант типа **FilterGroupEnum** (см. свойство Filter)).

Метод **Find** осуществляет поиск в наборе данных записи, соответствующей указанной строке критерия. Строка критерия должна содержать наименование только одного поля (например, "FName LIKE 'Jo*'"). Поиск по нескольким полям этот метод не поддерживает. Поиск обычно начинается с текущей позиции. Кроме того, можно указать смещение начальной позиции относительно текущей, а также направление поиска. Если текущая позиция не определена, то вызов метода **Find** влечет за собой ошибку.

Формат запуска метода Find следующий:

MyRec.Find Criteria, SkipRows, SearchDirection, Start

Строка **Criteria** содержит критерий поиска (оператор, содержащий имя поля, оператор сравнения и значение). Необязательный аргумент **SkipRows** типа Long задает смещение начальной позиции поиска от текущей записи (или от стартовой закладки, если она была задана в аргументе **Start**). Значение по умолчанию для этого аргумента — **0**. Необязательный аргумент **SearchDirection** задает направление поиска. Его значением может быть одна из двух констант типа **Search Direction Enum**: **adSearchForward** (направление поиска — вперед; в случае неудачи поиск останавливается в конце набора данных, а свойство **EOF** приобретает значение **True** (Истина)), **adSearchBackward** (направление поиска — назад (в направлении начала набора); в случае неудачи поиск останавливается в начале набора данных, а свойство **BOF** приобретает значение **True** (Истина)). В качестве значения последнего аргумента метода **Find** можно указать закладку, которая будет стартовой позицией для поиска. Не забудьте проверить перед этим, поддерживает ли набор записей применение закладок (для этого используется метод **Supports** с аргументом **adBookmark**). Ниже приведен пример итерации по всем клиентам, код которых начинается с букв **QU**. Каждый новый поиск начинается с позиции, непосредственно следующей за позицией записи, найденной предыдущим поиском.

```
MyRec.Find "[КодКлиента] LIKE 'QU*' "
Do While MyRec.EOF<>True
  Debug.Print MyRec![КодКлиента]
  MyRec.Find "[КодКлиента] LIKE 'QO*' ", 1, adSearchForward
Loop
```

Метод Move перемещает указатель текущей записи в указанную позицию.

recordset.Move NumRecords, Start

Метод **Move** принимает два аргумента. Первый аргумент типа Long указывает количество записей, на которые необходимо передвинуть указатель либо относительно текущей записи, либо относительно позиции, заданной вторым аргументом. Если значение первого аргумента отрицательно, то указатель текущей позиции перемещается в сторону начала набора данных. Второй аргумент является необязательным и задает стартовую позицию для перемещения. В качестве значения этот аргумент может содержать либо закладку, либо одну из констант типа **BookmarkEnum**: **adBookmarkCurrent** (текущая позиция является стартовой), **adBookmarkFirst** (перемещение будет осуществляться от первой записи набора), **adBookmarkLast** (стартовой позицией будет последняя запись набора).

Группа методов **MoveFirst**, **MoveLast**, **MoveNext**, **MovePrevious**. При помощи этих методов можно перейти соответственно к первой, последней, следующей и предыдущей записи. Первые два метода используются для быстрого перехода в начало или конец набора данных, а остальные два используются для организации итерации по всему набору данных.

Метод **Requery** используется для обновления набора данных путем повторного выполнения запроса, лежащего в основе объекта **Recordset**. Метод **Requery** используется для того, чтобы заставить объект **Recordset** отображать все сделанные к текущему моменту изменения набора данных.

Метод **Save** сохраняет набор данных объекта **Recordset** в файле на диске или в объекте **Stream**.

recordset.Save Destination, PersistFormat

Метод принимает два аргумента: **Destination** и **PersistFormat**. Аргумент **Destination** задает имя и полный путь к файлу на диске или ссылку на объект **Stream**. Второй аргумент задает формат, в котором данные будут сохранены. В качестве его значения можно задать одну из двух констант типа **PersistFormatEnum**: **adPersistADTG** (данные будут сохранены в формате ADTG (Microsoft Advanced Data TableGram)) и **adPersistXML** (формат XML).

Пример использования объекта **Recordset.**

```
Set MyRec = New ADODB.Recordset
Set MyRec1 = New ADODB.Recordset
MyRec.Open "Фонд", CurrentProject.Connection, adOpenStatic, adLockOptimistic
MyRec1.Open "Фонд1", CurrentProject.Connection, adOpenStatic, adLockOptimistic
Set MyComm = New ADODB.Command
MyComm.ActiveConnection = CurrentProject.Connection
MyComm.CommandText = "Delete * from Фонд1;"
MyComm.CommandType = adCmdText
MyComm.Execute

While MyRec.EOF <> True
If (MyRec!Автор Like d + "*") Then
MyRec1.AddNew
MyRec1!Название = MyRec!Название
MyRec1!Автор = MyRec!Автор
MyRec1!Ном_номер = MyRec!Ном_номер
MyRec1.Update
End If
MyRec.MoveNext
Wend
MyRec.Close
MyRec1.Close
Set MyComm = Nothing
Set MyRec = Nothing
Set MyRec1 = Nothing
```

Объект Command

Объект **Command** используется для того, чтобы сформировать запрос к базе данных для возвращения необходимых объекту **Recordset** записей (при этом возвращается ссылка на временный набор данных) или манипуляции структурой самой базы данных. Объект **Command** удобно использовать для запуска выполняемых запросов (на добавление, удаления записей и других), а также хранимых на стороне сервера процедур. Вы можете сами ознакомиться с тем небольшим количеством свойств и методов, имеющихся у объекта **Command**.

Последовательность действий, выполняемых при использовании объекта **Command**, во многом повторяет группу действий, которые необходимо выполнить, чтобы открыть объект **Recordset**. Схематично эту последовательность действий можно описать примерно так (конечно, это не единственно возможная схема):

Описать переменную типа **Connection** (например, MyConn) и открыть соединение (так как это описывалась ранее в этой главе).

1. Описать переменную типа Command.

Dim MyComm AS ADODB.Command

2. Явно создать экземпляр объекта:

Set MyComm = NEW ADODB.Command

3. Связать созданный объект с открытым ранее соединением:
MyCommand.ActiveConnection = MyConn

4. Указать тип и текст команды. Для этого используются свойства **CommandType** и CommandText. В приведенном ниже отрывке кода в качестве текста команды указывается текст SQL-оператора и соответствующий тип команды:
MyComm.CommandText = "Delete * from [Временная];"
MyComm.CommandType = adCmdText

5. При необходимости задать значения других свойств, например, время ожидания завершения команды **CommandTimeout**.
6. Осталось только выполнить команду при помощи метода **Execute**: **MyComm.Execute**. В зависимости от типа выполняемой команды есть некоторые различия в использовании метода **Execute**. Эти различия поясняются в приведенных ниже примерах.

Использование объекта Command для запуска запроса на выполнение

В этом примере создается объект **MyComm** типа **Command**. При помощи свойства **ActiveConnection** объект **Command** связывается с открытым соединением (объектом типа **Connection**). В качестве текста команды указывается строка SQL-оператора, предназначенного для удаления всех записей из таблицы **Временная**, а в качестве значения свойства **CommandType** задается константа **adCmdText**. В заключение примера освобождается память и ресурсы, занятые объектами.

```
Dim MyConn As ADODB.Connection
Dim ConStr As String
Dim MyComm AS ADODB.Command

ConStr="Provider = Microsoft.Jet.OLEDB.4.0; Data Source = C:\db2.mdb"
Set MyConn= New ADODB.Connection
MyConn.Open ConStr

Set MyComm = New ADODB.Command
MyCommand.ActiveConnection = MyConn
MyComm.CommandText = "Delete * from [Временная];"
MyComm.CommandType = adCmdText
MyComm. Execute

MyConn.Close
Set MyConn = Nothing
Set MyComm = Nothing
```

Использование объекта Command для возвращения набора записей

Начало этого примера напоминает начало предыдущего примера. Отличие состоит в следующем. В дополнение к другим объектам объявляется и создается экземпляр объекта типа **Recordset**. В качестве текста команды для объекта **MyComm** указывается имя таблицы **Временная**, а для свойства **CommandType** задается соответствующая константа **adCmdTable**. Обратите внимание на формат использования метода **Execute**. Поскольку в этом случае в результате его выполнения возвращается набор записей, то применена конструкция **SET**, инициализирующая переменную **MyRec**. В результате выполнения этой строки кода набор данных будет открыт, то есть отпадает необходимость использования

метода **Open** объекта **Recordset**. Для иллюстрации этого служит итерация по записям набора, организованная при помощи цикла **DO UNTIL ... LOOP**. В заключение примера освобождается память и ресурсы, занятые объектами.

```
Dim MyConn As ADODB.Connection
Dim MyRec As ADODB. Recordset
Dim ConStr As String
Dim MyComm AS ADODB.Command

ConStr="Provider = Microsoft.Jet.OLEDB.4.0; Data Source = C:\db2.mdb"
Set MyConn= New ADODB.Connection
Set MyRec = New ADODB.Recordset
MyConn.Open ConStr

Set MyComm = New ADODB.Command
MyCommand.ActiveConnection = MyConn
MyComm.CommandText = "[Временная];"
MyComm.CommandType = adCmdTable
Set MyRec = MyComm. Execute
Do Until MyRec.EOF
        Debug.Print MyRec![КодКлиента]
        MyRec.MoveNext
Loop
MyRec.Close
MyConn.Close
Set MyConn = Nothing
Set MyComm = Nothing
Set MyRec = Nothing
```

Примеры использования объекта Command

Пример 1

```
Dim res As New Collection
Dim conn As ADODB.Connection
Dim cmd As New ADODB.Command
Set conn = Access.CurrentProject.AccessConnection

cmd.ActiveConnection = conn
cmd.CommandText = "SELECT p.Name AS DependencyName FROM (Dependencies AS d
INNER JOIN Packages AS p ON d.DependencyID=p.PackageID) INNER JOIN Packages AS pd
ON pd.PackageID=d.DependentID WHERE d.DependentID = (SELECT PackageID from Packages
Where Name = @Name)"
cmd.Parameters("@Name").Value = packageName

Dim rs As ADODB.Recordset
Set rs = cmd.Execute
Do While Not rs.EOF
res.Add (rs("DependencyName").Value)
rs.MoveNext
Loop

conn.Close
```

Пример 2

```
Set MyComm = New ADODB.Command
MyComm.ActiveConnection = CurrentProject.Connection
MyComm.CommandText = "Delete * from Фонд1;"
MyComm.CommandType = adCmdText
MyComm.Execute
Set MyComm = Nothing
```

Как правильно сгенерировать значения ADO RecordCount

Свойство ADO **RecordCount** возвращает количество записей в наборе записей ADO. Однако в некоторых случаях данное свойство возвращает значение, равное -1. Это происходит из-за того что значение, возвращаемое свойством **RecordCount**, зависит от типа курсора в наборе записей: **-1** для курсора, перемещаемого только вперед; **реальное количество записей** — для статического курсора или курсора, управляемого клавишами на клавиатуре; **-1** или **реальное количество записей** — для динамического курсора в зависимости от источника данных.

Кроме того значение **RecordCount** равно **-1** для наборов записей, созданных с помощью метода **Execute** для объекта **Connection** или **Command**. Это происходит потому, что данный метод генерирует набор записей, перемещаться по которому можно только вперед. В примере -1 будет возвращено для набора записей на базе myConRst и **реальное количество записей** — для myKeyRst.

```
Sub TestRecordCount()
  Dim myConn As ADODB.Connection
  Dim myComm As String
  Dim myConRst As ADODB.Recordset
  Dim myKeyRst As ADODB.Recordset
  Dim sConnection As String
  sConnection = "Provider=Microsoft.Jet.OLEDB.4.0;" & _
        "Data Source=D:\Microsoft Visual Studio\VB98\Biblio.mdb"

  Set myConn = New ADODB.Connection
  Set myKeyRst = New ADODB.Recordset

  myConn.Open sConnection
  myComm = "Select * From Authors"
  Set myConRst = myConn.Execute(myComm, , adCmdText)

  myKeyRst.Open myComm, myConn, adOpenKeyset
  MsgBox "RecCount from Connection: " & myConRst.RecordCount & _
        vbCr & "From Recordset: " & myKeyRst.RecordCount

  Set myKeyRst = Nothing
  Set myConRst = Nothing
  Set myConn = Nothing

End Sub
```

В следующих заданиях использовать **Модель доступа к данным ADO.**

 1. Из собственной базы данных подключиться к базе Борей.mdb. Добавить в таблицу **Доставка** новую запись.

 2. В свою базу данных импортировать таблицы из базы Борей.mdb. В таблице **Товары** всем товарам с типом фрукты к имени товара (поле **Марка**) добавить строку “!!!” (использовать навигацию по записям).

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

1. Робинсон С. Microsoft ACCESS 2000: Учебный курс. – СПб.: Питер, 2001
2. Блюттман Кен, Фриз Уэйн. Анализ данных в Access. Сборник рецептов. – СПб.: Питер, 2008. – 352 с.
3. Сеннов Андрей Светозарович. Access 2007: Учебный курс. – СПб.: Питер, 2008. – 272с.
4. Вейскас Джон. Эффективная работа: Microsoft Office Access 2003. – СПб.: Питер, 2005. – 1168с.
5. Стив Ламберт: Microsoft Office Access 2007. Русская версия. М.: ЭКОМ, 2007. – 432с.

СОДЕРЖАНИЕ