Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Кафедра электронных вычислительных средств

А. В. Станкевич

СУБД Access

Лабораторный практикум
по курсу «Прикладные пакеты САПР проблемно-ориентированных электронных вычислительных средств»

для студентов специальности I-40 02 02
«Электронные вычислительные средства»
дневной формы обучения

Минск 2008

УДК 004.65 + 004.9 (075.8)
ББК 32.973.26-018.2 я 73
С 76

Р е ц е н з е н т
доцент кафедры ЭВМ БГУИР,
канд. техн. наук А. А. Петровский

Станкевич, А. В.
С 76 СУБД Access : лаб. практикум по курсу «Прикладные пакеты САПР проблемно-ориентированных электронных вычислительных средств» для студ. спец. I-40 02 02 «Электронные вычислительные средства» днев. формы обуч. / А. В. Станкевич. – Минск : БГУИР, 2008. – 55 с. : ил.
ISBN 978-985-488-303-8

В практикуме приведены описания лабораторных работ по дисциплине «Прикладные пакеты САПР проблемно-ориентированных электронных вычислительных средств» с использованием СУБД Access для студентов специальности I-40 02 02 «Электронные вычислительные средства» дневной формы обучения.

УДК 004.65 + 004.9 (075.8)
ББК 32.973.26-018.2 я 73

ISBN 978-985-488-303-8

Содержание

Лабораторная работа №1. Разработка структуры базы данных и таблиц 4
Лабораторная работа №2. Создание запросов к базе данных 18
Лабораторная работа №3. Создание форм и отчетов 37
Лабораторная работа №4. Автоматизация приложения с помощью модулей 44

ЛАБОРАТОРНАЯ РАБОТА №1

Разработка структуры базы данных и таблиц

Цель работы: научиться создавать с помощью системы управления базами данных (СУБД) Microsoft Access таблицы реляционной базы данных, устанавливать связи между таблицами, добавлять и изменять данные в таблицах.

1 Теоретические сведения

1.1 Основные понятия реляционных баз данных

Реляционные базы данных (БД) реализуют *реляционную модель данных*, в которой данные представляются в виде совокупности взаимосвязанных таблиц. Каждый объект такой совокупности описывается в виде таблицы, в которой хранятся свойства объекта.

Поле (атрибут) – свойство описываемого объекта, соответствующее столбцу таблицы.

Запись (кортеж) – совокупность значений атрибутов объекта, соответствующая строке таблицы.

Таблица (отношение) – совокупность записей с заполненными значениями полей.

Таблица реляционной БД должна удовлетворять следующим ограничениям:

- каждый элемент таблицы – один элемент данных, повторяющиеся группы данных должны отсутствовать;
- все столбцы в таблице должны быть однородными, т.е. все элементы в столбце должны иметь одинаковый тип данных (числовой, символьный и т.д.) и длину;
- каждый столбец должен иметь уникальное имя;
- одинаковые строки в таблице должны отсутствовать.

Для обеспечения уникальности и идентификации каждой записи в таблице используется *первичный ключ*. Первичный ключ состоит из набора значений, которые однозначно определяют строку таблицы. Любому значению первичного ключа должна соответствовать только одна строка таблицы. Первичный ключ включает только одно поле в том случае, если это поле не содержит повторяющихся значений.

Если для выполнения условий, накладываемых на значения первичного ключа, этот ключ включает несколько полей таблицы, то такой ключ называется *составным*.

Внешний ключ – столбец, значения которого соответствуют значениям первичного ключа другой связанной таблицы. Такой ключ может состоять как из одного, так и из нескольких полей (составной внешний ключ). Если число полей внешнего ключа меньше, чем количество полей соответствующего первичного ключа, то он называется частичным или усеченным внешним ключом.

Чтобы связать две реляционные таблицы, необходимо ключ первой таблицы ввести в состав ключа второй таблицы (возможно совпадение ключей), либо нужно ввести в структуру первой таблицы внешний ключ, который будет соответствовать первичному ключу второй таблицы.

Между таблицами реляционной БД могут быть установлены следующие *типы связей (типы отношений)*.

Связь «один-к-одному». В этом случае каждая запись в первой таблице может иметь не более одной связанной записи во второй таблице и наоборот. Таблицы с таким типом связи могут быть объединены в одну. Обычно такой тип связи может использоваться для разделения таблиц, содержащих много полей, для отделения части таблицы по соображениям конфиденциальности или для сохранения сведений, относящихся к некоторому подмножеству записей.

Связь «один-ко-многим». В этом случае каждая запись в первой таблице может иметь более одной связанной записи во второй таблице. Это наиболее распространенный тип связи. Например, один автор из таблицы авторов книг может иметь более одной связанной записи в таблице со сведениями о книгах, если он является автором более одной книги.

Связь «многие-к-одному» является противоположной предыдущему типу связи, если смотреть на нее с противоположной стороны (со стороны таблицы, в которой многие связи могут соответствовать одной связи другой таблицы).

Связь «многие-ко-многим». В этом случае одной записи в первой таблице могут соответствовать несколько записей во второй таблице, а одной записи во второй таблице могут соответствовать несколько записей в первой таблице. Этот тип связи реализуется через третью промежуточную таблицу, первичный ключ которой состоит из внешних ключей, связанных отношениями «многие-к-одному» с первичными ключами первой и второй таблиц. Например, связь «многие-ко-многим» между таблицами «Авторы» и «Книги» реализуется путем создания двух связей «один-ко-многим» этих таблиц с таблицей «Авторы книг».

1.2 Проектирование базы данных

При проектировании БД необходимо решить следующие основные вопросы:

- отобразить объекты предметной области в объекты реляционной модели данных (поставить в соответствие реальным объектам таблицы, а свойствам объектов – поля таблицы);
- исключить избыточность хранимых данных;
- разработать такую структуру БД, которая позволит обеспечить эффективность выполнения запросов (минимизировать время выборки, обновления, удаления, добавления и сортировки данных).

При этом надо иметь в виду, что при решении двух последних вопросов к структуре БД могут быть предъявлены противоречивые требования и необходимо будет искать разумный компромисс.

При проектировании БД часто используется методология нормализации, под которой понимается разбиение одной таблицы на несколько новых, позволяющих исключить проблемы при обновлении, добавлении и удалении данных. Для большинства практических задач достаточно представление таблиц БД в третьей нормальной форме. В этом случае каждая таблица должна удовлетворять следующим требованиям [1]:

- удовлетворять ограничениям на таблицу реляционной БД;
- не содержать повторяющихся групп данных;
- данные во всех неключевых полях должны однозначно определяться первичным ключом и каждым полем составного первичного ключа;
- все неключевые поля таблицы должны зависеть от первичного ключа, но должны быть независимыми друг от друга.

Процесс проектирования БД рассмотрим на примере. Пусть необходимо разработать структуру БД, в которой хранятся данные о микросхемах и их корпусах. При этом необходимо учесть возможность наличия нескольких конструктивных исполнений (разных корпусов) для одной микросхемы и одинаковых корпусов для разных микросхем.

Анализ предметной области показывает, что имеется объект – микросхема, свойства которой должны быть включены в таблицу «Микросхемы». В данном случае примем во внимание следующие свойства, позволяющие отличать конкретную микросхему от других:

- обозначение;
- функциональное назначение;
- производитель.

Включать информацию о корпусе микросхемы в таблицу «Микросхемы» не имеет смысла, поскольку возникнут проблемы при решении вопроса о возможности выпуска одной микросхемы в разных корпусах. При хранении информации о корпусе в таблице «Микросхемы» пришлось бы дублировать данные об обозначении, функциональном назначении и производителе для каждого нового корпуса данной микросхемы. Помимо дополнительных затрат внешней памяти возникли бы проблемы при обновлении и удалении информации о микросхеме (пришлось бы просматривать всю таблицу и вносить изменения или удалять все найденные записи). Поэтому информацию о корпусе микросхемы вынесем в отдельную таблицу «Корпуса». В этой таблице будем хранить значения следующих свойств корпуса:

- обозначение корпуса;
- чертеж.

Определим первичные ключи этих таблиц. В качестве кандидата на роль первичного ключа в таблице «Микросхемы» может быть выбрано поле «Обозначение», однако если существует возможность использования одинакового обозначения разных микросхем разными производителями, то это поле может содержать повторяющиеся значения, что недопустимо для поля первичного ключа. С учетом данного обстоятельства введем дополнительное

поле «Код микросхемы», которое будет хранить порядковый номер записи о каждой добавляемой микросхеме, и сделаем это поле первичным ключом.

Для таблицы «Корпуса» в качестве первичного ключа выберем поле «Обозначение корпуса», полагая, что это обозначение уникально.

Из словесного описания задачи проектирования следует, что между таблицами «Микросхемы» и «Корпуса» существует связь типа «многие-ко-многим», которая реализуется через дополнительную таблицу. Введем таблицу «Корпуса микросхем», которая будет содержать поля «Код микросхемы» и «Обозначение корпуса», а также поле «Идентификатор», которое будет хранить обозначение микросхемы в конкретном корпусе. Установим связи «один-ко-многим» между первичными ключами таблиц «Микросхемы» и «Корпуса» и внешними ключами таблицы «Корпуса микросхем». В качестве первичного ключа таблицы «Корпуса микросхем» возьмем составной ключ из полей внешних ключей, поскольку он будет уникальным для каждой записи. Окончательная структура разработанной БД представлена на рисунке 1.

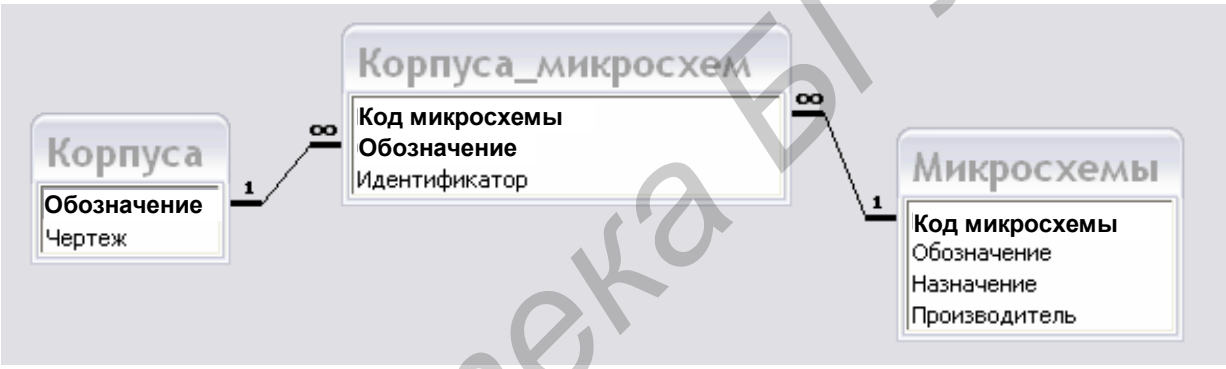

Рисунок 1 – Структура БД

На рисунке 1 полужирным шрифтом выделены поля первичных ключей, символом ∞ обозначена сторона связи «многие».

1.3 Создание новой базы данных и таблиц с использованием СУБД Access

При описании команд Access будет использоваться синтаксис ***Пункт меню/Команда/Подкоманда/…*** , который означает, что необходимо выбрать указанный пункт главного меню и далее последовательно в раскрывающихся меню выбрать требуемую команду, подкоманду и т.д.

Для создания новой базы данных запустите MS Access и выполните команду ***Файл/Создать***. В открывшейся панели инструментов выберите пункт ***Новая база данных***. Далее в открывшемся стандартном окне Windows создания файла укажите место размещения и имя новой БД. В результате выполнения команды откроется окно новой БД (рисунок 2).

В левой части окна представлены объекты Access.

Таблицы – предназначены для хранения данных БД.

Запросы – позволяют извлекать данные из таблиц, обновлять, добавлять и удалять данные в таблицах, создавать и изменять таблицы, работать со связями.

Формы – предназначены для упрощения работы пользователя с базой данных путем создания пользовательского интерфейса.

Отчеты – предназначены для вывода данных на печать в отформатированном виде, удобном для восприятия человеком.

Страницы – Web-страницы для удаленного доступа к БД.

Макросы – набор из одной или более макрокоманд, выполняющих последовательность повторяющихся действий.

Модули – код Visual Basic for Application (VBA) для автоматизации работы приложения БД.

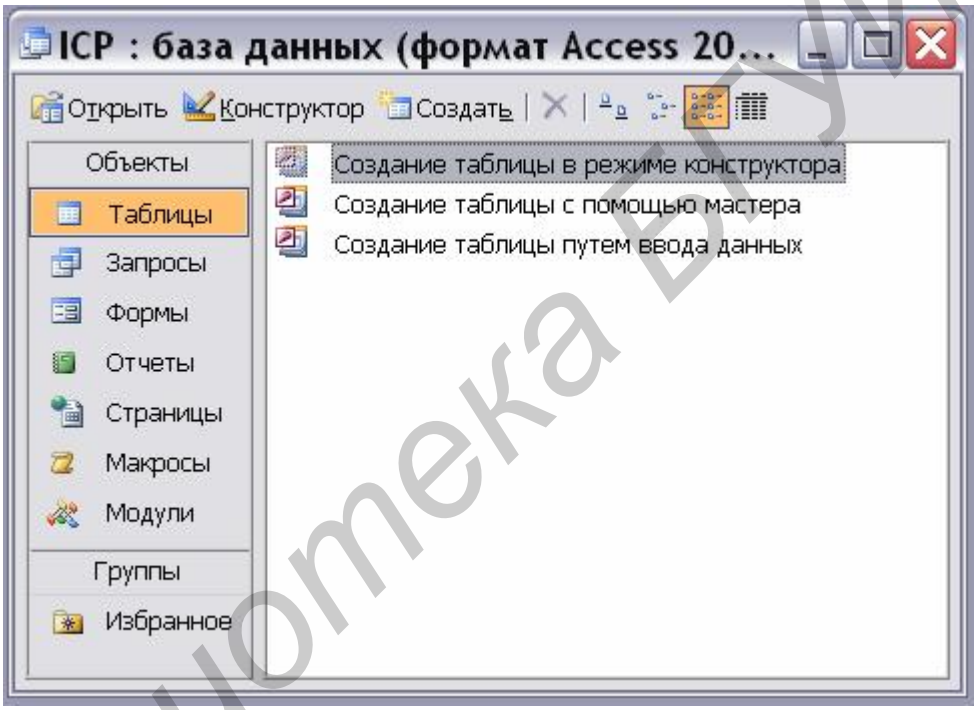

Рисунок 2 – Окно базы данных

Собственно к БД можно отнести таблицы и запросы. Остальные объекты Access являются объектами приложения БД.

Прежде чем создавать объекты Access, рассмотрим *соглашение об именах объектов Microsoft Access.* Имена объектов должны содержать не более 64 символов и могут включать любые комбинации букв, цифр, пробелов и специальных символов за исключением точки, восклицательного знака, апострофа (верхней одиночной кавычки) и квадратных скобок. Имя не должно начинаться с пробела. Лучше не включать в имена объектов пробелы, в особенности если предполагается часто использовать ссылки на эти имена в выражениях или в программе VBA, а также стараться избегать длинных имен, поскольку на них неудобно ссылаться.

Для создания новой таблицы необходимо в окне БД (см. рисунок 2) выделить объект **Таблицы**, после чего в правой части окна будут доступны варианты создания таблиц. Таблицу в Access можно создать следующими способами:

- использовать мастер для создания всей БД, содержащей требуемые отчеты, таблицы и формы, за одну операцию. Мастер создает новую БД, но его нельзя использовать для добавления новых таблиц, форм, отчетов в уже существующую БД;
- использовать мастер таблиц, позволяющий выбрать поля для данной таблицы из множества заранее определенных таблиц;
- ввести данные непосредственно в пустую таблицу в режиме таблицы (при сохранении таблицы каждому полю будет присвоен необходимый тип данных и формат);
- создать макет таблицы в режиме конструктора;
- импортировать данные из другого источника;
- создать таблицу с помощью инструкции SQL.

Поскольку любую таблицу можно далее редактировать в режиме конструктора, а также в связи с ограниченностью объема методического пособия рассмотрим только вариант создания таблицы в режиме конструктора. В этом случае в окне БД нужно выбрать вариант **Создание таблицы в режиме конструктора** (рисунок 2). После выбора данного варианта создания таблицы откроется окно макета таблицы (рисунок 3), в котором в бланке свойств задаются свойства полей таблицы:

- **Поле первичного ключа** (первый столбец в бланке, не имеющий имени). Для установки (удаления) первичного ключа необходимо выделить в бланке строку с именем требуемого поля и щелкнуть по кнопке с изображением ключа  . Для создания составного ключа необходимо производить выделение требуемых полей при нажатой клавише «Ctrl»;
- **Имя поля**. Должно удовлетворять соглашению об именах объектов;
- **Тип данных**. Определяет тип данных, сохраняемых в данном поле таблицы. Из поля со списком можно выбрать следующие типы данных:

§ *Текстовый* – текст или числа, не требующие проведения расчетов. Длина до 255 символов. Является типом данных по умолчанию;

§ *Поле MEMO* – текст длиной до 65535 символов (используется для примечаний);

§ *Числовой* – для числовых данных (имеются различные подтипы данных), позволяет проводить расчеты;

§ *Дата/время* – для хранения даты и времени;

§ *Денежный* – для числовых данных с фиксированной запятой, с точностью до 15 знаков в целой и до 4 знаков в дробной части;

§ *Счетчик* – уникальные последовательно возрастающие на единицу или случайные числа, автоматически вводящиеся в данное поле при добавлении каждой новой записи в таблицу. Значения полей данного типа пользователь изменить не может;

§ *Логический* – логические значения, а также поля, которые могут содержать одно из двух возможных значений (True/False, Да/Нет);

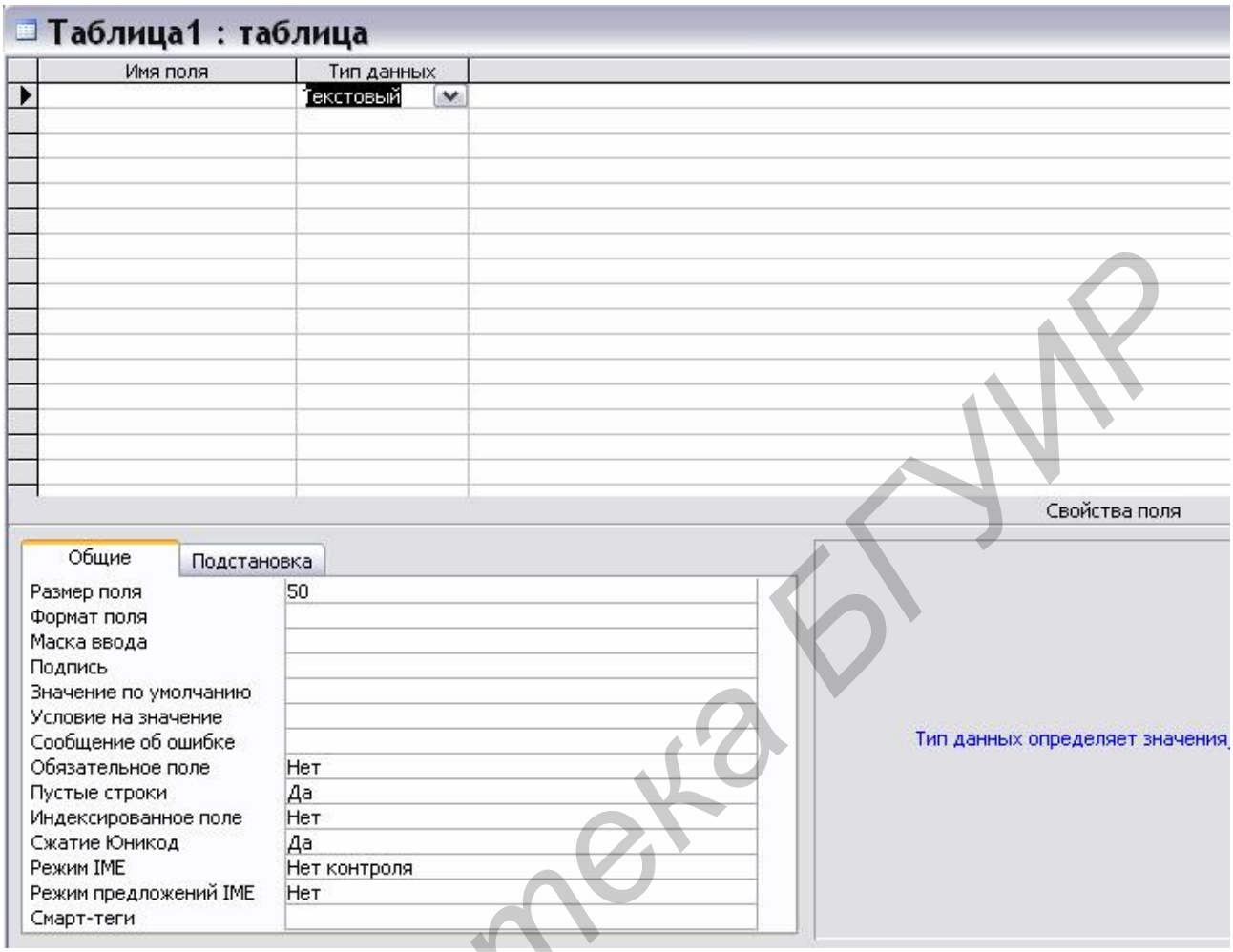

Рисунок 3 – Макет таблицы

§ *Поле объекта OLE* – объект, связанный или внедренный в таблицу Microsoft Access (Object Linking and Embedding of objects – технология связывания и внедрения объектов и протокол, разработанные компанией Майкрософт). Это может быть рисунок, видео или звуковой файл и т.д. Объем до 1 Гбайт (ограничивается доступным объемом внешней памяти);

§ *Гиперссылка* – адрес гиперссылки;

§ *Мастер подстановок* – позволяет создать поле, в котором предлагается выбор значений из списка или из поля со списком, содержащего набор постоянных значений или значений из другой таблицы.

- **Описание** – комментарий.

Остальные свойства поля заполняются в нижней части бланка. На вкладке **Подстановка** можно задать поведение поля в форме и таблице. Вкладка **Общие** содержит дополнительные свойства поля, которые отличаются для разных типов данных поля:

- **Размер поля**. Определяет максимальный размер данных (задается не для всех типов данных). Размеры полей:

§ Для *текстового поля* размер – число символов, не превышающее минимальное из двух значений: 255 или значение свойства **Размер поля**.

§ Для *числового поля*:

- *Байт.*Числа от 0 до 255 (без дробной части). Размер – 1 байт;
- *Целое.* Числа без дробной части от минус 32 768 до 32 767. Размер – 2 байта;
- *Длинное целое* (значение по умолчанию). Числа без дробной части от минус 2 147 483 648 до 2 147 483 647. Размер – 4 байта;
- *Одинарное с плавающей точкой.* Числа от минус 3,402823E38 до минус 1,401298E-45 для отрицательных значений и от 1,401298E-45 до 3,402823E38 для положительных. Дробная часть – 7 знаков, размер – 4 байта;
- *Двойное с плавающей точкой.* Числа от минус 1,79769313486232E308 до минус 4,94065645841247E–324 для отрицательных значений и от 4,94065645841247E–324 до 1,79769313486231E308 для положительных. Дробная часть – 15 знаков, размер – 8 байт;
- *Код репликации.* Уникальный глобальный идентификатор (GUID) для реплики. Размер – 16 байт;
- *Действительное.* Числа от минус 10^{-28}-1 до 10^{28}-1. Дробная часть – 28 знаков, размер – 12 байт.

- **Формат поля**. Задает формат вывода чисел и дат. Возможно использование встроенных и специальных пользовательских форматов, созданных при помощи символов форматирования. В Microsoft Access определены стандартные форматы для полей с типами данных «Числовой», «Дата/время», «Логический», «Текстовый» и «Поле MEMO». В качестве стандартных используются национальные форматы, выбираемые в окне **Язык и стандарты** панели управления Windows.
- **Число десятичных знаков**. Определяет число десятичных знаков после десятичной запятой, используемых для отображения чисел.
- **Маска ввода**. Задает строку символов, облегчающую ввод данных в поле. Например, удобно создать следующую маску ввода для поля «Телефон», позволяющую вводить только цифры и автоматически добавляющую промежуточные символы для междугороднего кода и разделитель групп цифр: (___) ___-____. Для создания маски ввода удобно использовать мастера по созданию масок ввода. Для запуска мастера необходимо щелкнуть по пиктограмме .
- **Подпись.** Указывает текст, выводимый в подписях к элементам управления и в заголовках столбцов таблицы вместо имен полей. В случае отсутствия подписи будет использовано имя поля.
- **Значение по умолчанию**. Позволяет задать значение, автоматически появляющееся в данном поле при создании новой записи.
- **Условие на значение**. Позволяет контролировать значения данных, вводимых в таблицу. Значение данного свойства определяется с помощью выражения, включающего любую комбинацию операторов, констант, функций, имен полей, элементов управления или свойств, результатом которой является

конкретное значение. Максимальная длина выражения в свойстве **Условие на значение** равна 2048 символов. Например >0.

- **Сообщение об ошибке**. Позволяет указать текст сообщения об ошибке, если введенные данные нарушают условие на значение (см. предыдущее свойство).
- **Обязательное поле**. Позволяет задать поля, в которых пустые значения Null не допускаются (при вводе новой записи поле должно быть обязательно заполнено). Пустое значение Null используется для представления отсутствующих или неизвестных данных. Значение Null может использоваться в выражениях и запросах.
- **Пустые строки**. Определяет, допускается ли в данное поле ввод пустых строк, не содержащих текстовых символов (функция Len для такой строки возвращает 0). Свойства **Обязательное поле** и **Пустые строки** позволяют различать несуществующие данные (сохраняемые в поле в виде пустых строк "") и данные, которые, возможно, существуют, но пока не известны (сохраняемые в виде пустых значений Null). Если свойство **Пустые строки** имеет значение «Да», пустые строки являются допустимыми значениями данного поля вне зависимости от значения свойства **Обязательное поле**. Если для свойства **Обязательное поле** задано значение «Да», а для свойства **Пустые строки** значение «Нет», требуется ввод в поле любого непустого значения, причем пустые строки не допускаются.
- **Индексированное поле**. Определяет индекс, создаваемый по данному полю. Допустимые значения: **Нет**; **Да, допускаются совпадения**; **Да, совпадения не допускаются**. Кроме задания данного свойства, можно выбрать команду ***Индексы*** в меню ***Вид*** или нажать кнопку  на панели инструментов. Будет открыто окно индексов, в котором можно сделать соответствующие установки. После определения индекса по одному полю в окне индексов свойство **Индексированное поле** автоматически примет значение «Да». Допускается создание произвольного количества индексов. Не допускается создание индексов для полей MEMO, гиперссылок и объектов OLE. Индексы, содержащие несколько полей (не более 10), следует определять в окне индексов. При этом имя индекса для разных полей должно быть одно. Поле первичного ключа индексируется автоматически. Соображения о необходимости индексации остальных полей таблицы приведены ниже.
- **Новые значения**. Определяет способ изменения значений поля счетчика при добавлении новых записей. «Последовательные» – значение поля счетчика увеличивается на 1 для каждой новой записи. «Случайные» – поле счетчика в новой записи получает псевдослучайное значение (для присвоения записям таблиц, содержащихся в различных репликах, уникальных идентификаторов).

Практические рекомендации по выбору типа данных

Следует задавать минимально возможное значение свойства **Размер поля**, поскольку в этом случае обработка данных выполняется быстрее и для хранения требуется меньше памяти.

При достаточной точности до четырех знаков в дробной части чисел лучше использовать денежный тип данных, поскольку в этом случае обработка данных выполняется быстрее за счет использования фиксированной запятой.

Следует иметь в виду, что сортировать, индексировать и делать ключевыми поля с типом данных MEMO, гиперссылки и объекты OLE невозможно.

При сортировке также нужно учитывать, что числа в текстовых полях сортируются как строки чисел (1, 10, 100, 2, 20, 200 и т.д.), а не как числовые значения.

Структура таблицы определяется путем задания имен и других свойств ее полей. После этого определяются ключевые поля таблицы (см. с.6). После определения структуры всех таблиц необходимо задать связи между таблицами. Связи в БД необходимы для выполнения запросов, которые работают с данными нескольких таблиц (многотабличных запросов).

Индексация полей

Индексы используются СУБД для привязки данных ключевых полей к их физическому положению на диске. Основное назначение индексов – ускорение доступа к записям или группам записей таблиц, что ускоряет выполнение запросов.

Access автоматически создает индекс по полю первичного ключа, кроме того, можно задать другие индексы. При этом надо иметь в виду следующее: индекс позволяет быстрее сортировать данные и не требует упорядочивать набор записей по индексу первичного ключа, что позволяет повысить скорость выполнения запросов. С другой стороны, при добавлении новой записи Access тратит время на обновление всех индексов таблицы.

Для повышения производительности приложений Access при работе с большими таблицами и запросами по этим таблицам нужно выполнять следующие практические правила:

- Минимизируйте число индексов в таблицах, используемых в транзакциях (транзакция – последовательность вносимых изменений), особенно в многопользовательских сетевых приложениях с совместным использованием таблиц. Пока редактируются записи и обновляются индексы, другой пользователь не может изменять таблицы.
- Минимизируйте число индексов в таблицах, которые регулярно используются запросами на добавление и удаление. Время, уходящее на обновление индексов, особенно заметно при изменении больших таблиц.
- Индексируйте поля внешних ключей таблиц, участвующих в соединениях с первичными таблицами. Однако при задании критерия выбора по ключевому полю всегда указывайте поле первичного ключа вместо поля внешнего ключа или поля связанной таблицы.
- Индексируйте поля, для которых устанавливаются критерии выбора.

Эффективность использования индексов проверяется по скорости работы запроса. Если скорость работы запроса существенно возросла, то проверьте

скорость выполнения транзакций. Работа с запросами будет рассмотрена в следующей лабораторной работе.

1.4 Создание связей в базе данных

Для создания связей необходимо в окне базы данных выполнить команду ***Сервис/Схема данных*** (кнопка кнопочной панели ). Откроется окно **Схема данных**. При первом вызове команды это окно будет пустым. Для добавления новой таблицы или запроса в схему данных нужно выполнить команду ***Связи/Добавить таблицу***, а для удаления таблицы необходимо выделить эту таблицу и нажать клавишу «Delete».

Для создания связей между двумя таблицами необходимо выбрать поле, по которому устанавливается связь, в первой таблице и перетащить его во вторую таблицу на то поле, с которым устанавливается связь. Для связывания сразу нескольких полей их перемещение осуществляется при нажатой клавише «Ctrl». Для удаления связи необходимо ее выделить и нажать клавишу «Delete».

Связываемые поля могут иметь разные имена, однако они должны иметь одинаковые типы данных, а для полей числового типа также должны быть одинаковыми значения свойства **Размер поля**. Из приведенного выше правила есть исключение: поле счетчика можно связывать с числовым полем с размером **Длинное целое**.

При установлении и редактировании связи (двойном щелчке левой кнопкой мыши по соответствующей связи) откроется окно **Изменение связей** (рисунок 4). В этом окне можно установить флажок **Обеспечение целостности данных**. Для связей, у которых определена целостность данных, пользователь имеет возможность указать, следует ли автоматически выполнять для связанных записей операции каскадного обновления и каскадного удаления.

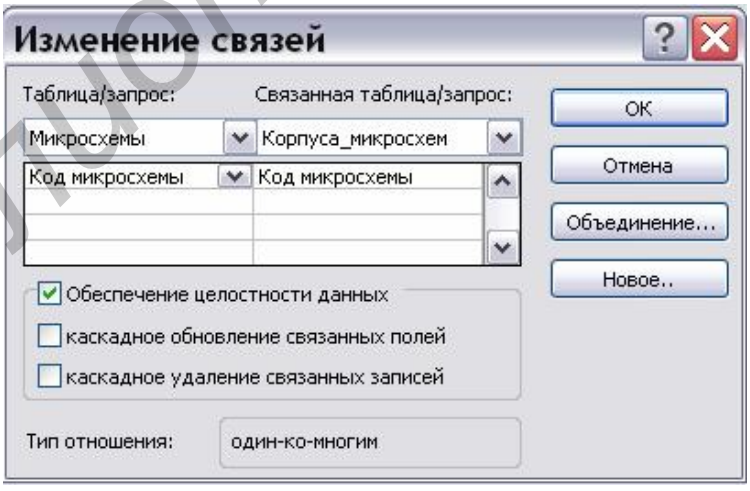

Рисунок 4 – Окно **Изменение связей**

Под целостностью данных в данном случае понимается ссылочная целостность, для которой необходимо, чтобы всем значениям внешних ключей соответствовали значения первичных ключей базовых таблиц.

Если для соответствующей связи установить флажок **Каскадное обновление связанных полей**, то любое изменение значения в ключевом поле главной таблицы приведет к автоматическому обновлению соответствующих значений во всех связанных записях. Например, при изменении значения поля «Обозначение» в таблице «Корпуса» будет автоматически без предупреждений обновлено поле «Обозначение» во всех записях таблицы «Корпуса микросхем».

Следует иметь в виду, что если в главной таблице ключевым полем является поле с типом данных **Счетчик**, то установление флажка **Каскадное обновление связанных полей** не приведет к каким-либо результатам, так как нельзя изменить значение поля счетчика.

Если для соответствующей связи установить флажок **Каскадное удаление связанных записей**, то удаление записи в главной таблице приведет к автоматическому удалению связанных записей в подчиненной таблице. При этом Microsoft Access выводит предупреждение о возможности удаления связанных записей.

1.5 Работа с таблицами в СУБД Access

Добавление новых данных в режиме таблицы

1 Откройте таблицу в режиме таблицы (в окне базы данных выберите объект **Таблицы** и двойным щелчком левой кнопки мыши откройте требуемую таблицу).

2 Нажмите на панели инструментов в нижней части открывшегося окна кнопку  . Курсор будет установлен на первое поле новой записи.

3 Введите необходимые данные и нажмите клавишу «Tab» для перехода к следующему полю.

4 В конце записи нажмите клавишу «Tab» для перехода к следующей (новой) записи.

Редактирование данных таблицы осуществляется установкой курсора в требуемую ячейку с последующим изменением существующих значений.

Сортировка, поиск и фильтрация данных в таблицах

Сортировка может осуществляться по возрастанию или убыванию по одному полю или по нескольким полям. Для группового выделения нескольких полей должна использоваться клавиша «Shift». После выделения требуемых полей используется команда ***Записи/Сортировка/Сортировка по возрастанию*** или ***Записи/Сортировка/Сортировка по убыванию*** (кнопка  или  соответственно).

Поиск по значению в поле осуществляется командой ***Правка/Найти***. В открывшемся окне задается искомое сочетание символов. Следует иметь в виду, что поиск по началу поля или по всему полю работает быстрее, чем по любой части поля.

Для замены используется команда ***Правка/Заменить***.

Фильтрация данных в таблице

Фильтрация используется для отображения подмножества записей исходной таблицы, соответствующего заданному условию. Имеются различные виды фильтров.

Фильтр по выделенному. Для создания такого фильтра необходимо выделить требуемое значение в ячейке таблицы и выполнить команду ***Записи/Фильтр/Фильтр по выделенному***.

Обычный фильтр. Для создания такого фильтра используется команда ***Записи/Фильтр/Изменить фильтр***. В бланке фильтра для соответствующего поля из списка значений выбирается требуемое. Если нужно объединять по ИЛИ несколько значений поля, то необходимо выбрать закладку ИЛИ внизу бланка фильтра. Для разных полей таблицы критерии отбора объединяются по И. Для применения фильтра необходимо выполнить команду ***Фильтр/Применить фильтр***. Этой же командой применяется любой вид фильтра.

Расширенный фильтр. Создается с помощью команды ***Записи/ Фильтр/Расширенный фильтр.*** В бланке фильтра в строке **Поле** выбираются требуемые поля таблицы, а в строке **Условие отбора** вводятся критерии отбора. В строках **или** при необходимости вводятся другие критерии отбора для данного поля. Условия отбора для разных полей одной строки объединяются по И.

Для удаления любого вида фильтра, независимо от того, каким способом он был создан, необходимо переключиться в окно расширенного фильтра и выполнить команду ***Правка/Очистить бланк***, после чего применить фильтр.

2 Порядок выполнения работы

1 Изучить порядок разработки базы данных, последовательность создания структуры таблицы базы данных Access, типы данных и свойства полей таблицы, порядок создания связей между таблицами и настройку параметров связей.

2 Получить задание у преподавателя.

3 Разработать структуру базы данных и таблиц для выполнения требований выданного задания.

4 С помощью Access в соответствии с полученным заданием создать требуемые таблицы и установить связи на схеме данных.

5 В каждой таблице заполнить не менее семи записей для возможности последующего выполнения запросов по этим таблицам и проверки получаемых результатов.

6 Оформить отчет по лабораторной работе.

3 Содержание отчета

1 Название работы и цель работы.
2 Исходное задание.
3 Макеты таблиц базы данных, схема данных.
4 Выводы.

4 Контрольные вопросы

1 Какими свойствами обладает первичный ключ?

2 Для чего используется индексация полей таблицы?

3 Какие требования должны быть выполнены для поддержания ссылочной целостности в базе данных?

4 Для чего используется свойство поля «Условие на значение»?

5 Из каких соображений выбирается тип данных полей таблицы?

6 Каким образом создать связь между полями таблиц и как установить параметры этой связи?

7 Поля с каким типом данных можно индексировать?

8 Какие требования предъявляются к полям, по которым устанавливается связь между таблицами?

9 Для чего используются и как создаются маски ввода?

10 Для чего устанавливаются связи между таблицами?

11 Чем отличаются свойства поля «Обязательное поле» и «Пустые строки»?

Литература

1 Дженнингс, Р. Microsoft Access 97 в подлиннике. Т. 2 / Р. Дженнингс. – СПб.: BHV – Санкт-Петербург, 1997.

2 Праг, К. Access 2002. Библия пользователя / К. Праг, М. Ирвин. – М. : Диалектика-Вильямс, 2003.

3 Система помощи Microsoft Access 2003.

ЛАБОРАТОРНАЯ РАБОТА №2

Создание запросов к базе данных

Цель работы: научиться создавать с помощью Access различные типы запросов к базе данных.

1 Теоретические сведения

1 Основные сведения о запросах к базе данных

Запросы предназначены для выборки данных по заданным критериям из одной или нескольких таблиц, изменения, добавления и удаления записей таблиц, создания и модификации структуры таблиц, работы с индексами, связями и первичными ключами.

Имеется несколько типов запросов. Наиболее часто используемым является *запрос на выборку*. Запрос на выборку используется в следующих целях:

- получение требуемых данных из одной или нескольких таблиц;
- группировка записей для вычисления сумм, средних значений и других статистических и групповых функций;
- изменение (с некоторыми ограничениями) данных в таблицах.

В зависимости от количества таблиц, на базе которых может быть построен запрос на выборку, запросы делятся на однотабличные и многотабличные.

Запрос с параметрами – это запрос, при выполнении которого открывается диалоговое окно для ввода параметров (например для ввода условий отбора записей).

Перекрестный запрос позволяет сгруппировать записи по двум наборам данных, один из которых расположен в левом столбце таблицы (заголовки строк), а второй – в верхней строке (заголовки столбцов). В ячейках получившейся таблицы приводятся результаты статистических или групповых расчетов (суммы, количество записей, средние значения и т.п.).

Запрос на изменение – это запрос, который за одну операцию вносит изменения в несколько записей. Существует четыре вида запросов на изменение: на удаление, обновление записей, добавление записей и создание таблицы.

Запрос на удаление удаляет группу записей из одной или нескольких таблиц, причем можно удалять только всю запись, а не отдельные поля внутри нее.

Запрос на обновление записей позволяет изменять данные группы записей в существующих таблицах.

Запрос на добавление позволяет добавить в конец одной или нескольких таблиц группу записей из одной или нескольких таблиц.

Запрос на создание таблицы позволяет создать новую таблицу на основе всех или части данных из одной или нескольких таблиц. Чаще всего такой

запрос используется для создания резервной копии требуемых данных и создания таблицы для экспорта в другую базу данных.

Запрос SQL – это запрос, создаваемый при помощи инструкций SQL. Язык SQL (Structured Query Language – структурированный язык запросов) используется при создании запросов, а также для обновления и управления реляционными базами данных, такими как базы данных Microsoft Access.

Варианты создания запросов:

- с помощью мастера запросов создается запрос путем ответов пользователя на поставленные вопросы. Мастера можно также использовать для быстрого создания структуры запроса, а для его доработки использовать режим конструктора;
- использование режима конструктора и бланка запроса по образцу QBE (Query by Example);
- ввод инструкции SQL в окно SQL;
- создание запросов на основе фильтра при сохранении фильтра как запроса.

Более подробно рассмотрим создание запроса в режиме конструктора и в окне SQL.

1.2 Создание и работа с запросами на выборку

Для создания любого типа запроса, в том числе и запроса на выборку, необходимо в окне базы данных выбрать объекты **Запросы** (см. рисунок 2). В правой части окна можно выбрать один из двух вариантов: **Создание запроса в режиме конструктора** или **Создание запроса с помощью мастера**. Рассмотрим работу в режиме конструктора как наиболее часто используемую.

При выборе этого варианта создания запроса откроется бланк QBE (рисунок 5).

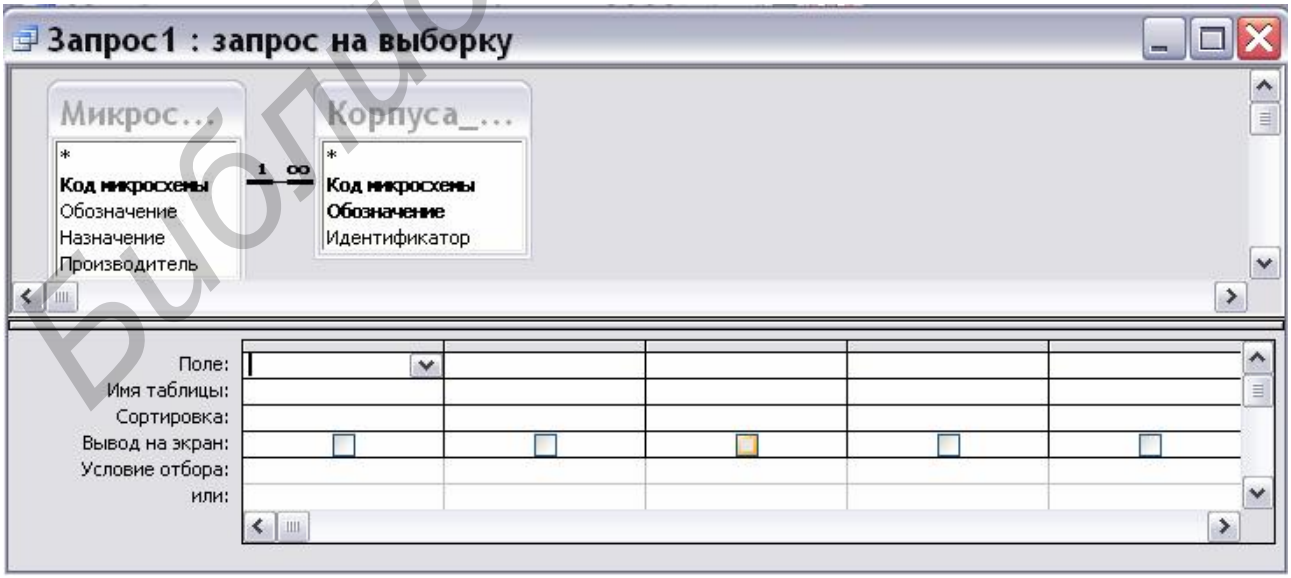

Рисунок 5 – Окно запроса по образцу

Для добавления таблицы или запроса в бланк запроса по образцу необходимо выполнить команду ***Запрос/Отобразить таблицу***. Для добавляемых таблиц автоматически будет показана связь между таблицами. В бланке запроса также можно определить связи между таблицами путем перетаскивания связываемых полей, а через контекстное меню либо через двойной щелчок по связи изменить параметры объединения.

В нижней части бланка QBE в строке **Поле** из списка выбирается поле требуемой таблицы, которое должно присутствовать в запросе. Также можно перетащить в эту строку имя поля соответствующей таблицы из верхней части бланка. Для включения в запрос всех полей таблицы в каждой таблице самой первой строкой является строка, содержащая символ «*». Эту строку следует перетащить в нужный столбец строки **Поле** нижней части бланка.

В строке **Сортировка** можно при необходимости выбрать из поля со списком сортировку по возрастанию или убыванию.

Флажок в строке **Вывод на экран** позволяет указать необходимость отображения данного поля в результирующем множестве записей после выполнения запроса. Например, какое-то поле может участвовать в запросе для условия отбора, но отображать его не нужно.

Условие отбора задает критерий отбора для соответствующего поля.

Строка **или** позволяет для одного поля задать несколько условий, объединяемых по ИЛИ.

Условия, введенные в одной строке для разных полей, объединяются по И.

Когда пользователь создает запрос в режиме конструктора запроса, Microsoft Access автоматически создает эквивалентную инструкцию SQL. Пользователь имеет возможность просматривать и изменять инструкции SQL в режиме SQL. Для переключения в режим SQL надо выполнить команду ***Вид/Режим SQL***. Изменения, внесенные в запрос в режиме SQL, приведут к соответствующим изменениям в бланке запроса в режиме конструктора и наоборот. Следует иметь в виду, что не все виды запросов можно представить в режиме конструктора.

1.3 Выражения в запросах

Выражения в запросах используются в качестве условия отбора, а также для создания вычисляемых полей.

Выражение представляет комбинацию операторов, литералов, констант, функций и идентификаторов, позволяющую получить определенное значение.

Операторы – это обычные арифметические и другие операторы Access.

Литералом называют значение в явном представлении, например, число, строковое значение или дата. Даты и время необходимо заключать в символы (#), а строковые значения в двойные кавычки ("). Например "PQ208" или #1-янв-07#.

Константа представляет неизменяющееся значение. Например, в Access определены константы True, False и Null (пустое значение).

Функция возвращает значение, которое является результатом расчетов. Пользователь имеет возможность создавать собственные функции на языке VBA.

Идентификатор представляет ссылку на значение поля, элемента управления или свойства.

Операторы

Арифметические операторы: +; -; *; /; ^ - степень;\ – деление нацело (11 \ 4 возвращает результат 2); Mod – остаток от деления нацело (11 Mod 4 возвращает 3) . При делении нацело в случае, если любой из операндов является действительным числом, он предварительно округляется до целого (11.5\4=3).

Операторы присваивания и сравнения: <;<=;>;>=;=;<>.

Логические операторы используются чаще всего для объединения нескольких выражений в одно. Наиболее часто в выражениях используются операторы логического И (And) двух выражений, логического ИЛИ (Or) и логического отрицания (Not).

Оператор слияния строк (конкатенации) & позволяет объединить в одну несколько текстовых строк.

Оператор Is чаще всего используется в выражениях вместе с константой Null для проверки значения (например Is Null или Is Not Null).

Оператор Like используется для сравнения строкового выражения с образцом. Имеет следующий синтаксис:

[Not] Like "образец".

Для аргумента *образец* можно задавать полное значение (например Like "Иванов") или использовать подстановочные знаки для поиска диапазона значений.

Подстановочные знаки используются в выражениях, чтобы включить значения, соответствующие определенному образцу. Можно использовать следующие подстановочные знаки:

* – соответствует любому количеству символов и может быть использован в любом месте текстовой строки (ст* позволяет найти «стол», «стул» , «стакан»; *ка позволяет найти «лампочка», «папка», «краска»);

? – соответствует любому одиночному символу (л?па позволяет найти «лапа», «липа» , «лупа»);

– соответствует любой одиночной цифре (1#3 позволяет найти 103, 113, 123);

[ ] – соответствует любому символу, расположенному в квадратных скобках (л[аи]па позволяет найти «лапа» и «липа», но не «лупа»);

! – соответствует любому символу, не включенному в список (л[!аи]па находит «лупа», но не «лапа» и не «липа»);

- – соответствует любому символу из диапазона символов (б[а-в]д находит «бад», «ббд» и «бвд»).

Подстановочные знаки * (звездочка), ? (вопросительный знак), # (знак числа) и [(открывающая квадратная скобка) рассматриваются как образец для поиска этого символа только при заключении их в квадратные скобки.

Оператор In позволяет проверить, совпадает ли значение выражения с одним из элементов указанного списка. Синтаксис оператора:

[Not] In(значение_1; значение_2; . . .).

Например In("Москва"; "Минск"; "Киев").

Для русского языка в качестве разделителя списка используется символ «;» вместо «,» в английской версии. Поэтому в бланке запроса должны стоять символ «;» а в выражениях SQL – «,». Также необходимо помнить, что для русского языка разделителем дробной части является «,», а не «.», соответственно в бланке запроса и других визуальных средствах нужно использовать символ «,», а в SQL и VBA – «.».

Оператор Between позволяет определить принадлежность значения выражения указанному диапазону. Синтаксис оператора:

[Not] Between значение_1 And значение_2.

Если значение поля, для которого определено это условие, попадает в диапазон, задаваемый аргументами *значение_1* и *значение_2* (включительно), оператор Between возвращает значение True, в противном случае возвращается значение False. Логический оператор Not позволяет проверить противоположное условие (что выражение находится за пределами диапазона, заданного с помощью аргументов *значение_1* и *значение_2*). Чаще всего речь идет о числовых значениях, однако оператор работает и для текстовых данных. Например Between #04/1/99# And #07/1/99#. Для проверки принадлежности к диапазону можно использовать операторы >,<.

Функции

Access имеет большое количество встроенных функций. Рассмотрим только статистические функции, используемые для определения статистических данных на основе наборов числовых значений в запросах с группировкой.

Функция Avg(выражение) вычисляет среднее арифметическое набора чисел, содержащихся в указанном поле запроса. Здесь и далее при описании статистических функций аргумент *выражение* является строковым выражением, которое определяет поле, содержащее числовые данные для вычисления среднего значения, или выражение, выполняющее вычисления с данными из этого поля. Операнды аргумента *выражение* могут включать имя поля таблицы, константу или функцию. Функция может быть внутренней или определяться пользователем, но не одной из других статистических функций SQL.

Функции First(выражение), Last(выражение) возвращают значение поля из первой или последней записи результирующего набора запроса.

Функции Min(выражение), Max(выражение) возвращают минимальное и максимальное значения из набора значений, содержащихся в указанном поле запроса.

Функции StDev(выражение), StDevP(выражение) возвращают смещенное и несмещенное значение среднеквадратичного отклонения, вычисляемого по набору значений, содержащихся в указанном поле запроса.

Функции Var(выражение), VarP(выражение) возвращают значение смещенной и несмещенной дисперсии, вычисляемой по набору значений, содержащихся в указанном поле запроса.

Функция Count(выражение) вычисляет количество записей, возвращаемых запросом.

Функция Sum(выражение) возвращает сумму набора значений, содержащихся в заданном поле запроса.

Имена объектов в выражениях

При вводе имен объектов (таблиц, форм, отчетов, запросов, полей и элементов управления) в идентификаторы использование квадратных скобок, в которые заключается имя, является обязательным, если имя содержит пробелы или специальные символы, например, символ подчеркивания. Имена, не содержащие пробелы и специальные символы, можно вводить без квадратных скобок.

Иногда возникает необходимость ссылки в выражении на поле или другой элемент управления формы. В этом случае синтаксис ссылки имеет следующий вид:

Forms!ИмяФормы!ИмяЭлементаУправления.

Например, для использования в качестве условия отбора значения из поля «Обозначение» формы «Микросхемы» необходимо сделать следующую ссылку:

Forms![Микросхемы]![Обозначение]

Построитель выражений

При создании сложных выражений целесообразно пользоваться построителем выражений. Для его вызова необходимо в поле условия отбора бланка запроса по образцу из контекстного меню выбрать команду ***Построить*** или нажать кнопку  кнопочной панели. С помощью Построителя можно выбрать объект базы данных, функцию, оператор и т.д. При этом будет обеспечено правильное построение идентификаторов.

Использование окна отладки

При проверке значений, возвращаемых выражениями, целесообразно пользоваться окном отладки. Для вызова окна отладки в окне базы данных необходимо выбрать объекты **Модули** и с помощью кнопки **Создать** создать фиктивный модуль (его можно будет закрыть без сохранения). Далее необходимо выполнить команду ***View/Immediate Window***. Для проверки выражения необходимо в окне **Immediate** ввести символ «?» и проверяемое выражение и нажать клавишу «Enter». Например, для получения текущей даты можно воспользоваться функцией Date. В этом случае в окне **Immediate** нужно ввести ?Date и нажать клавишу «Enter». Следует помнить об отличиях в синтаксисе для визуальных средств и VBA.

Создание вычисляемых полей в запросах

Для создания вычисляемого поля в запросе необходимо в бланке запроса по образцу в строку **Поле** ввести имя вычисляемого поля и через двоеточие – выражение, по которому будет вычислено его значение. Например, в таблице имеются поля «Ток» и «Напряжение», тогда для вычисления мощности можно в бланке запроса создать следующее поле:

[Потребляемая мощность]:[Напряжение]*[Ток]

1.4 Запрос с параметрами

Если запрос предполагается выполнять многократно с изменением условий отбора, то целесообразно создать запрос с параметрами. Параметры можно добавлять в разные типы запросов. Запрос с параметрами отображает одно или несколько диалоговых окон, выводящих приглашение пользователю ввести условия отбора.

В режиме конструктора для каждого поля, которое предполагается использовать как параметр, необходимо ввести в ячейку строки **Условие отбора** текст приглашения пользователю, заключенный в квадратные скобки. Это приглашение будет выводиться в диалоговом окне при запуске запроса. Текст подсказки должен отличаться от имени поля, но может его включать. Например, для поля, в котором хранятся даты, можно вывести приглашения "Введите начальную дату:" и "Введите конечную дату:" для определения диапазона отбираемых значений по дате. Для этого необходимо в столбце данного поля бланка запроса по образцу в строке **Условие отбора** ввести выражение

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

Параметры по умолчанию имеют текстовый тип данных. Если условия отбора относятся к данным другого типа (в приведенном примере – к типу дата/время), то необходимо определить тип данных для параметров. Для этого необходимо выполнить команду ***Запрос/Параметры***. В столбец **Параметр** через буфер обмена необходимо вставить текст приглашения без квадратных скобок. Текст в строке условия отбора бланка запроса по образцу должен точно совпадать с содержимым столбца **Параметр**. В столбце **Тип данных** нужно задать требуемый тип.

Следует иметь в виду, что если Access не распознал какой-то идентификатор или другой элемент выражения в бланке запроса по образцу, то этот элемент чаще всего интерпретируется как параметр.

1.5 Групповые операции в запросах (запросы с группировкой)

Групповые операции используются для обобщения данных и группировки записей по значениям выбранных полей. Для выполнения групповых операций необходимо в режиме конструктора выполнить команду ***Вид/Групповые операции***. В бланке запроса появится дополнительная строка **Групповая операция**. Из раскрывающегося списка можно выбрать одну из статистических функций либо следующие дополнительные операции:

- **Группировка** – определяет группы, для которых выполняются вычисления. Например, чтобы посчитать общее количество микросхем в одинаковых корпусах, необходимо выбрать группировку по типу корпуса;
- **Выражение** – позволяет создать вычисляемое поле с помощью выражения, включающего статистическую функцию. Обычно вычисляемое поле создается, если требуется включить в выражение несколько функций;
- **Условие** – определяет условия отбора для поля, которое не участвует в группировке. Если для поля выбирается этот параметр, автоматически снимается флажок **Вывод на экран**, и поле не выводится на экран при выполнении запроса.

1.6 Перекрестные запросы

Перекрестные запросы обычно используются для получения итоговой информации. Рассмотрим построение перекрестного запроса в режиме конструктора на следующем примере. Необходимо сгруппировать данные по фирмам – производителям компонентов и по типам корпусов и рассчитать количество различных микросхем в одинаковых корпусах по производителям.

После включения в запрос требуемых полей таблиц «Микросхемы», «Корпуса» и «Корпуса_микросхем» необходимо выполнить команду ***Запрос/Перекрестный***. При этом в бланке запроса появятся строки **Групповая операция** и **Перекрестная таблица**. Бланк такого запроса по образцу представлен на рисунке 6. Необходимо определить заголовки строк и столбцов формируемой таблицы, а также значения ячеек этой таблицы так, как приведено на рисунке 6. Количество микросхем (значения ячеек) будем рассчитывать по полю «Код микросхемы» таблицы «Корпуса микросхем» с помощью статистической функции Count.

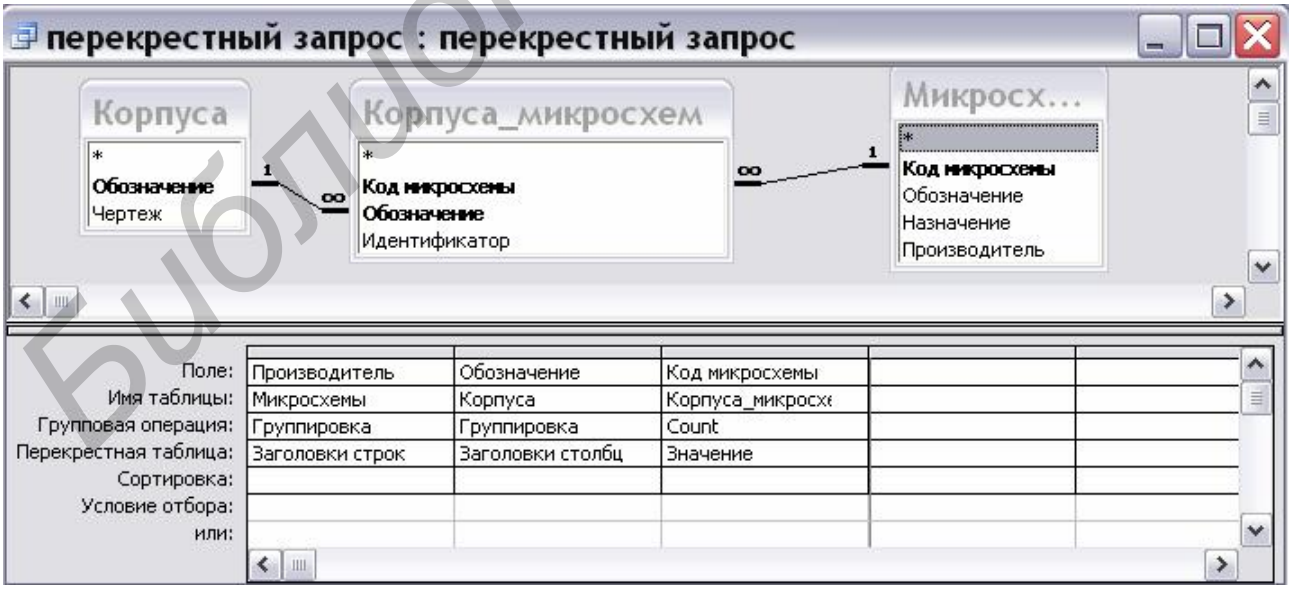

Рисунок 6 – Перекрестный запрос

1.7 Запросы на изменение

При создании запроса на изменение можно предварительно создать запрос на выборку для отбора данных из таблиц по требуемым критериям, а затем запрос на выборку преобразовать в соответствующий тип запроса на изменение, либо можно сразу создавать запрос на изменение, а перед его выполнением использовать для просмотра результатов команду ***Вид/Режим таблицы***. Рассмотрим виды запросов на изменение.

Запрос на создание таблицы

В режиме конструктора запроса необходимо выполнить команду ***Запрос/Создание таблицы***. Далее в открывшемся диалоговом окне задать имя таблицы и указать базу данных, в которой создается таблица. При выполнении запроса Access запросит подтверждение помещения записей в создаваемую таблицу. Следует иметь в виду, что свойства полей, кроме типа данных и размера, и ключевые поля исходной таблицы не переносятся в новую таблицу.

Запрос на добавление

В режиме конструктора запроса необходимо выполнить команду ***Запрос/Добавление***. Далее в открывшемся диалоговом окне задать имя таблицы, в которую добавляются данные, и указать базу данных, где находится эта таблица. Из списка полей в бланк запроса переместите с помощью мыши поля, которые необходимо добавить, а также те, которые будут использованы при определении условия отбора. Если в исходной таблице и в таблице, куда добавляются данные, поля имеют одинаковые имена, то имя автоматически вводится в строку **Добавление**. Если имена полей двух таблиц отличны друг от друга, то в строке **Добавление** необходимо указать требуемое имя поля. Следует иметь в виду, что запрос на добавление может включать меньше полей, чем таблица, в которую заносится информация.

Запрос на удаление записей

В режиме конструктора запроса необходимо выполнить команду ***Запрос/Удаление***, после чего в бланк запроса по образцу будет добавлена строка **Удаление**. В этой строке для удаляемых полей таблицы (обычно в этом случае сразу выбираются с помощью символа «*» все поля таблицы) указывается значение **Из**, а для полей, участвующих в формировании условий отбора записей, указывается значение **Условие**.

Поскольку отменить данную операцию невозможно, то прежде чем выполнить данный запрос, необходимо просмотреть выбранные для удаления данные (команда ***Вид/Режим таблицы***), а также желательно сделать резервную копию данных.

Следует иметь в виду, что при наличии связей между таблицами и опции каскадного удаления, помимо записей на стороне «один», удаляются также и связанные записи на стороне «многие».

Запрос на обновление

В режиме конструктора запроса необходимо выполнить команду ***Запрос/Обновление***, после чего в бланк запроса по образцу будет добавлена строка **Обновление**. Для полей, которые необходимо обновить, в строку

Обновление следует ввести выражение или значение, которое должно быть использовано для изменения полей. Для внесения обновлений необходимо выполнить запрос.

1.8 Запросы SQL и их использование

Некоторые запросы не могут быть определены в бланке запроса по образцу (запрос на объединение, управляющие запросы и некоторые другие). Для создания таких запросов требуется ввести инструкцию SQL непосредственно в окно запроса в режиме SQL. Для открытия окна SQL в режиме конструктора запроса необходимо выполнить команду ***Вид/Режим SQL***.

Запрос SQL реализуется с помощью соответствующей инструкции языка SQL. Каждая инструкция SQL должна заканчиваться символом «;» непосредственно за последним символом инструкции. Для разделения имени таблицы и имени поля в инструкциях используется точка. Имена полей и таблиц, которые содержат недопустимые символы (пробелы, зарезервированные символы), необходимо заключать в квадратные скобки.

Создание запросов на выборку (инструкция SELECT)

Упрощенный синтаксис инструкции SELECT следующий (ключевые слова языка выделены прописными буквами, символ «|» обозначает выбор одного из перечисленных вариантов, в квадратные скобки заключаются необязательные элементы языковой конструкции):

*SELECT [предикат] * | таблица.* | [таблица.]поле*
[AS псевдоним] [, [таблица.]поле [AS псевдоним] [, ...]]
FROM ИменаТаблиц [, ...]
[WHERE...]
[GROUP BY...]
[HAVING...]
[ORDER BY...];

Рассмотрим аргументы инструкции SELECT:

* – указывает, что необходимо выбрать все поля заданной таблицы или таблиц;

таблица – имя таблицы, из которой должны быть отобраны записи;

поле – имена полей, из которых должны быть отобраны данные. При включении в запрос несколько полей, они будут извлекаться в указанном порядке;

псевдоним – заголовки столбцов результата вместо исходных имен столбцов в таблице;

ИменаТаблиц – имена одной или нескольких таблиц, которые содержат отбираемые данные. Имена приводятся через запятые.

Предложение WHERE – определяет, какие записи из таблиц, перечисленных в предложении *FROM*, следует включить в результат выполнения запроса. Предложение имеет следующий синтаксис:

WHERE условиеОтбора

Здесь *условиеОтбора* – выражение, которому должны удовлетворять записи, включаемые в результат выполнения запроса.

Например, можно отобрать всех студентов университета от 18 до 20 лет:

WHERE Возраст Between 18 And 20

Предложение ***WHERE*** может содержать до 40 выражений, связанных логическими операторами And или Or. Если не задавать предложение ***WHERE***, запрос возвращает все строки исходной таблицы.

Предложение ***GROUP BY*** – группирует записи с одинаковыми значениями для указанного списка полей в одну запись. Если инструкция ***SELECT*** содержит статистическую функцию SQL, то для каждой записи будет вычислено итоговое значение с использованием этой функции. Предложение имеет следующий синтаксис:

GROUP BY группируемыеПоля

Здесь *группируемыеПоля* – имена полей, которые используются для группировки записей. Порядок имен полей (не более 10) определяет уровень группировки для каждого из этих полей.

При использовании предложения ***GROUP BY*** все поля в списке полей инструкции ***SELECT*** должны быть либо включены в предложение ***GROUP BY***, либо использоваться в качестве аргументов статистической функции SQL.

Предложение ***HAVING*** – определяет, какие сгруппированные записи отображаются при использовании инструкции ***SELECT*** с предложением ***GROUP BY***. После того как записи будут сгруппированы с помощью предложения ***GROUP BY***, предложение ***HAVING*** отберет те из полученных записей, которые удовлетворяют условиям отбора, указанным в предложении HAVING. Синтаксис предложения:

GROUP BY группируемыеПоля [HAVING условиеГруппировки]

Здесь *условиеГруппировки* – выражение, определяющее, какие сгруппированные записи следует отображать для столбца статистической функции в бланке запроса по образцу. После того как записи будут сгруппированы с помощью предложения ***GROUP BY***, предложение ***HAVING*** указывает, какие из полученных записей должны быть отобраны.

В отличие от предложения ***WHERE***, которое используется для исключения записей перед группировкой, предложение ***HAVING*** применяется для отбора записей после группировки.

Например, отберем из таблицы «Комплектующие» микросхемы, имеющиеся на складе в количестве более 10 штук:

SELECT КодМикросхемы, Sum(НаСкладе) AS Сумма FROM Комплектующие GROUP BY КодМикросхемы HAVING Sum(НаСкладе) > 10;

Предложение ***ORDER BY*** – сортирует записи, полученные в результате запроса, в порядке возрастания или убывания по значениям заданных полей. Имеет следующий синтаксис:

[ORDER BY поле_1 [ASC | DESC][, поле_2 [ASC | DESC]][, ...]]

Здесь *поле_1, поле_2* – имена полей, по которым сортируются записи.

Ключевые слова *ASC* или *DESC* задают порядок сортировки по возрастанию или убыванию для данного поля. Если эти ключевые слова опущены, то по умолчанию используется порядок сортировки по возрастанию (от "А" до "Я" и от 0 до 9).

В приведенном ниже примере отбираются фамилии сотрудников с сортировкой по убыванию окладов:

SELECT Фамилия, Оклад FROM Сотрудники ORDER BY Оклад DESC;

Предикат – используется для ограничения числа возвращаемых записей. В качестве предикатов могут быть: *ALL, DISTINCT, DISTINCTROW* или *TOP*.

Если предикат отсутствует, по умолчанию используется предикат *ALL*. В последнем случае отбираются все записи, соответствующие условиям, заданным в инструкции SQL.

Предикат DISTINCT позволяет исключать записи, которые содержат повторяющиеся значения в отобранных полях. Чтобы запись была включена в результат выполнения запроса, значения в каждом поле, включенном в инструкцию *SELECT*, должны быть уникальными. Например, в таблице «Студенты» есть однофамильцы. Если две записи содержат значение «Иванов» в поле «Фамилия», то следующая инструкция SQL возвратит только одну из них:

SELECT DISTINCT Фамилия FROM Студенты;

Если предложение *SELECT* содержит более одного поля и используется предикат *DISTINCT*, то для включения записи в результат выполнения запроса необходимо, чтобы совокупность значений во всех этих полях была уникальной.

Предикат DISTINCTROW позволяет исключить данные, основанные на целиком повторяющихся записях, а не на отдельных повторяющихся полях. Предикат *DISTINCTROW* влияет на результат только в том случае, если в запрос включены не все поля из анализируемых таблиц и запрос содержит более одной таблицы.

Предикат TOP n [PERCENT] позволяет возвратить определенное число записей, находящихся в начале или в конце диапазона, описанного с помощью предложения *ORDER BY*. Следующая инструкция SQL позволяет получить список 10 лучших студентов выпуска 2005 года:

SELECT TOP 10 Имя, Фамилия FROM Студенты
WHERE ГодВыпуска = 2005 ORDER BY СреднийБалл DESC;

Если в предыдущем примере средние баллы десятого и одиннадцатого студента будут равны, то запрос возвратит 11 записей.

Зарезервированное слово *PERCENT* используется для возврата определенного процента записей, находящихся в начале или в конце диапазона, описанного с помощью предложения *ORDER BY*. Например, необходимо отобрать 10 % худших студентов:

SELECT TOP 10 PERCENT Имя, Фамилия FROM Студенты
WHERE ГодВыпуска = 2005 ORDER BY СреднийБалл ASC;

Создание с помощью SQL многотабличных запросов

Многотабличные запросы строятся с использованием реляционной операции соединения.

Операция INNER JOIN (внутреннее соединение) – соединяет записи из двух таблиц, если поля этих таблиц, по которым установлена связь, содержат одинаковые значения. Операцию *INNER JOIN* можно использовать в любом предложении *FROM*. Операция имеет следующий синтаксис:

FROM таблица_1 INNER JOIN таблица_2 ON таблица_1.поле_1 оператор таблица_2.поле_2

Аргументы операции:

таблица_1, таблица_2 – имена таблиц, записи которых подлежат соединению;

поле_1, поле_2 – имена объединяемых полей. Если эти поля не являются числовыми, то они должны иметь одинаковый тип данных, однако поля могут иметь разные имена.

оператор – любой оператор сравнения: «=», «<», «>», «<=», «>=» или «<>», но чаще всего используется оператор «=».

Следующая инструкция SQL соединяет таблицы «Микросхемы» и «Корпуса_микросхем» по полю «Код микросхемы»:

SELECT Микросхемы.Обозначение, Корпуса_микросхем.Идентификатор
FROM Микросхемы INNER JOIN Корпуса_микросхем ON Микросхемы.[Код микросхемы] = Корпуса_микросхем.[Код микросхемы];

Операции *JOIN* могут быть вложенными; в этом случае, например, для четырех связанных таблиц, следует использовать следующий синтаксис:

SELECT поля
FROM таблица_1 INNER JOIN
(таблица_2 INNER JOIN (таблица_3)
INNER JOIN таблица_4 ON таблица_3.поле_3 оператор таблица_4.поле_4)
ON таблица_2.поле_2 оператор таблица_3.поле_2)
ON таблица_1.поле_1 оператор таблица_2.поле_1;

Операции LEFT JOIN, RIGHT JOIN – соединяют записи исходных таблиц в предложении *FROM* с помощью левого внешнего или правого внешнего соединения. Синтаксис операций:

FROM таблица_1 [LEFT | RIGHT] JOIN таблица_2
ON таблица_1.поле_1 оператор таблица_2.поле_2

Аргументы операций имеют тот же смысл, что и для операции *INNER JOIN*.

При левом внешнем объединении все записи из первой (левой) таблицы включаются в динамический результирующий набор, даже если во второй (правой) таблице нет соответствующих им записей.

При правом внешнем объединении все записи из второй (правой) таблицы включаются в динамический набор, даже если в первой (левой) таблице нет соответствующих им записей.

Операции ***LEFT JOIN*** или ***RIGHT JOIN*** могут быть вложены в операцию ***INNER JOIN***, но операция ***INNER JOIN*** не может быть вложена в операцию ***LEFT JOIN*** или ***RIGHT JOIN***.

Создание с помощью SQL запросов с параметрами

Для этой цели используется описание ***PARAMETERS***, которое описывает имя и тип данных каждого параметра запроса. Описание имеет следующий синтаксис:

PARAMETERS *имя типДанных [, имя типДанных [, ...]]*

Аргументы описания:

имя – имя параметра, являющегося строкой, которая отображается в окне диалога при выполнении запроса (не должно совпадать с именами полей таблиц). Строку, содержащую пробелы и знаки препинания, необходимо заключить в квадратные скобки.

типДанных – один из типов данных SQL ядра Microsoft Jet.

Описание ***PARAMETERS*** является необязательным, однако если оно присутствует, то должно находиться перед всеми остальными инструкциями, в том числе перед инструкцией ***SELECT***.

Следующая инструкция SQL запрашивает у пользователя производителя микросхем:

```
PARAMETERS [Введите производителя микросхем] Text;
SELECT Микросхемы.Обозначение, Микросхемы.Производитель
FROM Микросхемы
WHERE Микросхемы.Производитель=[Введите производителя микросхем];
```

Создание с помощью SQL перекрестных запросов

Для этой цели используется инструкция ***TRANSFORM***, имеющая следующий синтаксис:

```
TRANSFORM статФункция
SELECT
PIVOT выражение [IN (значение_1[, значение_2[, ...]])];
```

Аргументы инструкции ***TRANSFORM***:

статФункция – статистическая функция SQL, обрабатывающая указанные данные;

выражение – выражение, которое определяет заголовки столбцов для результирующего набора;

значение_1, значение_2 – фиксированные значения заголовков столбцов.

Инструкция *TRANSFORM* является необязательной, однако если она присутствует, то должна находиться впереди инструкции *SELECT*, которая указывает поля, содержащие заголовки строк, и предложения *GROUP BY*, которое задает группировку по строкам. При необходимости можно включить и другие предложения, например *WHERE*, для описания дополнительных условий отбора и сортировки. Если в запросе используются параметры, то описание *PARAMETERS* должно быть перед инструкцией *TRANSFORM*.

Следующая инструкция SQL реализует перекрестный запрос, представленный на рисунке 6:

```
TRANSFORM Count(Корпуса_микросхем.[Код микросхемы]) AS [Count-Код микросхемы]
SELECT Микросхемы.Производитель
FROM Микросхемы INNER JOIN (Корпуса INNER JOIN Корпуса_микросхем
ON Корпуса.Обозначение = Корпуса_микросхем.Обозначение) ON
Микросхемы.[Код микросхемы] = Корпуса_микросхем.[Код микросхемы]
GROUP BY Микросхемы.Производитель
PIVOT Корпуса.Обозначение;
```

Создание с помощью SQL запросов на изменение

Запрос на удаление. Реализуется с помощью инструкции *DELETE*, имеющей следующий синтаксис:

```
DELETE [таблица.*]
FROM таблица
WHERE условиеОтбора;
```

Аргументы инструкции *DELETE*:

таблица – имя таблицы, из которой удаляются записи;

условиеОтбора – выражение, определяющее удаляемые записи.

Запрос на удаление удаляет записи целиком, а не только содержимое указанных полей. Чтобы удалить данные в конкретном поле, следует создать запрос на обновление, который заменяет имеющиеся значения на значения Null.

Если в предложении *FROM* соединяются две или более таблицы, то можно удалять записи из таблицы со стороны «многие», или из одной из таблиц со стороны «один» при связи «один-к-одному».

Пример удаляения записей о всех микросхемах фирмы Xilinx.

```
DELETE * FROM Микросхемы WHERE Микросхемы.Производитель = 'Xilinx';
```

Запрос на добавление. Реализуется с помощью инструкции *INSERT INTO*. Упрощенный синтаксис инструкции на добавление нескольких записей:

```
INSERT INTO назначение SELECT [источник.]поле_1[, поле_2[, ...]
FROM выражение;
```

Синтаксис запроса на добавление одной записи:

```
INSERT INTO назначение [(поле_1[, поле_2[, ...]])]
VALUES (значение_1[, значение_2[, ...]);
```

Аргументы инструкции INSERT INTO:

назначение – имя таблицы, в которую добавляются записи;

источник – имя таблицы, откуда копируются записи;

поле_1, поле_2 – имена полей для добавления данных, если они следуют за аргументом *назначение*; имена полей, из которых берутся данные, если они следуют за аргументом *источник*;

выражение – имена таблиц, откуда вставляются данные. Это выражение может быть именем отдельной таблицы или результатом операции *INNER JOIN*, *LEFT JOIN* или *RIGHT JOIN*, а также сохраненным запросом;

значение_1, значение_2 – значения, добавляемые в указанные поля новой записи. Каждое значение будет вставлено в поле, занимающее то же положение в списке: значение_1 вставляется в поле_1 в новой записи, значение_2 – в поле_2 и т.д. Каждое значение текстового поля следует заключать в одиночные кавычки (' '); для разделения значений необходимо использовать запятые. Нужно определить все поля записи, в которые будет помещено значение, и значения для этих полей. Если поля не определены, в недостающие столбцы будет вставлено значение по умолчанию или значение Null.

Если результирующая таблица содержит ключ, убедитесь, что в ключевые поля добавляются уникальные непустые значения, в противном случае ядро базы данных Microsoft Jet не будет добавлять записи.

Запрос на создание таблицы. Реализуется с помощью инструкции *SELECT…INTO*, имеющей следующий синтаксис:

SELECT поле_1[, поле_2[, ...]] INTO новаяТаблица
FROM источник;

Аргументы инструкции *SELECT...INTO*:

поле_1, поле_2 – имена полей, которые следует скопировать в новую таблицу.

новаяТаблица – имя создаваемой таблицы.

источник – имя существующей таблицы или запроса, из которой отбираются записи.

При создании таблицы поля в новой таблице наследуют типы данных и размеры базовых полей, никакие другие свойства таблиц и полей не передаются.

Запрос на обновление. Реализуется с помощью инструкции *UPDATE*, имеющей следующий синтаксис:

UPDATE таблица
SET новоеЗначение
WHERE условиеОтбора;

Аргументы инструкции *UPDATE*:

таблица – имя таблицы, данные в которой следует изменить;

новоеЗначение – выражение, определяющее значение, которое должно быть вставлено в указанное поле обновленных записей; если таких полей несколько, они указываются через запятую;

условиеОтбора – выражение, отбирающее записи, которые должны быть изменены. При выполнении этой инструкции будут изменены только записи, удовлетворяющие указанному условию. В условии отбора можно ссылаться только на столбцы обновляемой таблицы.

Одновременно можно изменить значения нескольких полей. Следующая инструкция SQL увеличивает потребляемую мощность на 10 % для фирмы Xilinx в таблице «Параметры_микросхем»:

UPDATE [Параметры_микросхем]
*SET ПотрМощность = ПотрМощность * 1.1*
WHERE Производитель = 'Xilinx';

Запрос на объединение. Объединяет результаты нескольких независимых запросов или таблиц. Реализуется с помощью операции *UNION*, имеющей следующий синтаксис:

[TABLE] запрос_1 UNION [ALL] [TABLE] запрос_2 [UNION [ALL] [TABLE] запрос_n [...]]

Аргументы операции *UNION*:

запрос_1 – запрос_n – инструкция *SELECT*, имя сохраненного запроса или имя сохраненной таблицы (в последнем случае перед именем таблицы должно стоять зарезервированное слово *TABLE*).

В одной операции *UNION* можно объединить в любом наборе результаты нескольких запросов, таблиц и инструкций *SELECT*.

При использовании операции *UNION* по умолчанию повторяющиеся записи не возвращаются, однако в нее можно добавить предикат *ALL* для возврата всех записей.

Все инструкции *SELECT* или запросы (таблицы) должны возвращать одинаковое число полей в том же порядке. Соответствующие поля должны иметь совместимые типы данных за одним исключением: допускается объединение в одном поле значений полей типа «Числовой» и «Текстовый».

В запросах на объединение именами столбцов становятся имена полей из первой таблицы или из первой инструкции *SELECT*. Для переименования полей в результирующей таблице следует использовать предложение *AS*.

Подчиненный запрос. Подчиненным запросом называют инструкцию *SELECT*, вложенную в инструкцию *SELECT*, *SELECT...INTO*, *INSERT...INTO*, *DELETE* или *UPDATE* или в другой подчиненный запрос. Чтобы определить новое поле, данную инструкцию можно ввести в строку **Поле** в бланке запроса по образцу после определения имени поля. Чтобы указать для данного поля условие отбора, инструкция *SELECT* вводится в строку **Условие отбора**.

Варианты синтаксиса подчиненного запроса:

сравнение [ANY | ALL | SOME] (SELECT…)
выражение [NOT] IN (SELECT…)

[NOT] EXISTS (SELECT…)

Аргументы подчиненного запроса:

сравнение – выражение и оператор сравнения, который сравнивает выражение с результатами подчиненного запроса;

выражение – выражение, для которого проводится поиск в результирующем наборе записей подчиненного запроса;

SELECT… – инструкция *SELECT* (собственно подчиненный запрос), которая соответствует формату и всем правилам, принятым для инструкций *SELECT*. Инструкция должна быть заключена в круглые скобки.

Подчиненный запрос можно использовать вместо выражения в списке полей инструкции *SELECT* или в предложениях *WHERE* и *HAVING*.

Предикаты *ANY* или *SOME* являются синонимами и используются для отбора записей в главном запросе, которые удовлетворяют сравнению с любой записью, отобранной в подчиненном запросе.

Предикат *ALL* используется для отбора в главном запросе только тех записей, которые удовлетворяют сравнению со всеми записями, отобранными в подчиненном запросе.

Предикат *IN* используется для отбора в главном запросе только тех записей, которые содержат значения, совпадающие с одним из отобранных подчиненным запросом значений.

Предикат *NOT IN* используется для отбора в главном запросе только тех записей, которые содержат значения, не совпадающие ни с одним из отобранных подчиненным запросом.

Предикат *EXISTS* (с необязательным зарезервированным словом NOT) используется в логическом выражении для определения того, должен ли подчиненный запрос возвращать какие-либо записи.

Управляющие запросы. Данный тип запроса создает или вносит изменения в таблицы базы данных и используется для описания полей таблиц, индексов, создания и удаления таблиц, ключевых полей. В практикуме эти запросы не рассматриваются.

2 Порядок выполнения работы

1 Изучить типы запросов, порядок создания запросов на выборку данных, запросов с параметрами, с группировкой, перекрестных, запросов на изменение данных, SQL–запросов.

2 Получить задание у преподавателя.

3 С помощью Access в соответствии с полученным заданием создать требуемые запросы.

4 Оформить отчет по лабораторной работе.

3 Содержание отчета

1 Название работы и цель работы.

2 Исходное задание.

3 Копии бланков запросов по образцу или инструкции SQL для всех запросов, результаты выполнения запросов.

4 Выводы.

4 Контрольные вопросы

1 В чем отличие запросов на добавление от запросов на обновление?

2 Почему в запросах с группировкой должны использоваться статистические функции?

3 Как создать вычисляемое поле в запросе?

4 Как с помощью запроса можно создать таблицу?

5 Можно ли использовать запросы для изменения данных в таблицах?

6 Что такое подчиненный запрос?

7 Как можно повысить скорость выполнения запросов на выборку данных?

8 Для чего используются параметры в запросах?

9 Какие виды запросов не могут быть созданы с помощью бланка запроса по образцу QBE?

Литература

1 Дженнингс, Р. Microsoft Access 97 в подлиннике. Т. 2 / Р. Дженнингс. – СПб. : BHV – Санкт-Петербург, 1997.

2 Кузнецов, А. Microsoft Access 2003. Русская версия / А. Кузнецов. – СПб. : Питер, 2006.

3 Система помощи Microsoft Access 2003.

ЛАБОРАТОРНАЯ РАБОТА №3

Создание форм и отчетов

Цель работы: научиться создавать с помощью Access формы и отчеты приложения базы данных.

1 Теоретические сведения

1.1 Формы

С помощью форм реализуется пользовательский интерфейс в приложении базы данных. Как и другие объекты Access, форма создается в режиме конструктора или с помощью мастера. Форма состоит из окна и набора элементов управления и создается на базе таблицы или запроса.

После создания формы могут быть выведены на экран в следующих видах: режим конструктора, режим формы (окно с элементами управления), в режиме таблицы (отображается сразу столько записей, сколько может разместиться на экране), в режиме сводной таблицы и режиме диаграммы (используются для анализа данных и подведения итогов).

Параметры формы устанавливаются с помощью ее свойств. Для вызова окна свойств необходимо выполнить команду ***Вид/Свойства*** (команда открывает окно свойств выделенного объекта).

Окно свойств имеет следующие вкладки:

- **Макет** – позволяет задать внешний вид окна формы;
- **Данные** – определяет источник данных и возможности по изменению данных;
- **События** – предназначена для выбора событий в форме для подключения макросов и модулей;
- **Другие** – определяет тип окна, вид меню, наличие справки;
- **Все** – объединяет все свойства формы.

Рассмотрим некоторые *основные свойства формы*. С остальными можно ознакомиться, используя систему помощи Access.

Подпись (Caption) – заголовок окна формы.

Режим по умолчанию (DefaultView) – определяет режим открытия формы (простая форма, в которой отображается одна полная запись; ленточная форма, в которой отображается столько полных записей, сколько помещается в области данных формы; таблица; сводная таблица; сводная диаграмма).

Полосы прокрутки (ScrollBars) – определяет вывод полос прокрутки в форме.

Область выделения (RecordSelectors) – определяет вывод для формы области выделения записей в режиме формы.

Кнопки перехода (NavigationButtons) – определяет вывод в форме кнопок перехода по записям и поля номера записи.

Разделительные линии (DividingLines) – определяет вывод разделительных линий между разделами формы или между записями в ленточной форме.

Автоматический размер (AutoResize) – определяет при открытии окна формы автоматический выбор его размеров, достаточных для полного отображения записей.

Тип границы (BorderStyle) – позволяет указать тип и элементы границы, используемые в форме. Как правило, используются разные типы границ для обычных форм, всплывающих форм и специальных диалоговых окон. Возможные значения: **отсутствует** (форма не имеет границ и связанных с границей элементов, изменение ее размеров невозможно), **тонкая** (форма выводится с тонкой границей и может иметь любые элементы границы, изменение размеров формы невозможно), **изменяемая** (форма имеет стандартную границу форм Microsoft Access, может иметь любые из связанных с границей элементов и допускает изменение размеров, это значение по умолчанию), **окно диалога** (форма имеет границу двойной ширины и может включать строку заголовка, кнопку закрытия и оконное меню).

Источник записей (RecordSource) – определяет в качестве источника данных для формы таблицу, запрос или инструкцию SQL.

Разрешить изменение (AllowEdits), **Разрешить удаление (AllowDeletions)**, **Разрешить добавление (AllowAdditions)** – определяют, имеет ли пользователь возможность изменять, удалять или добавлять записи через форму.

Ввод данных (DataEntry) – определяет режим открытия формы, присоединенной к источнику данных, только для ввода данных. Для того чтобы открыть форму для ввода данных (с пустой записью), выберите значение **Да**. Для вывода в форме всех записей выберите значение **Нет**.

Тип набора записей (RecordsetType) – определяет тип набора записей формы. Значения свойства: **динамический набор** (значение по умолчанию, допускается изменение присоединенных элементов управления в единственной таблице или в нескольких таблицах со связями типа «один-к-одному», не допускается изменение данных для элементов управления, присоединенных к полям таблиц со связями типа «один-ко-многим» со стороны «один», если не разрешено каскадное изменение между таблицами); **динамический набор (несогл.)** (допускается редактирование всех таблиц и элементов управления, присоединенных к их полям); **статический набор** (не допускаются изменения данных в таблицах и в присоединенных к их полям элементах управления).

Всплывающее окно (PopUp) позволяет указать открытие формы в виде всплывающего окна над всеми остальными окнами Microsoft Access.

Модальное окно (Modal) позволяет указать открытие формы в режиме модального окна, при котором для перевода фокуса на другой объект необходимо предварительно закрыть форму.

Форма состоит из следующих разделов: заголовка, области данных и примечания. В ней также могут присутствовать верхний и нижний колонтитулы.

Пользователь имеет возможность разработать форму самостоятельно в режиме конструктора или создать форму с помощью мастера. Выбор варианта создания осуществляется в окне базы данных после выбора объектов **Формы**.

Элементы управления

Все элементы управления по типу можно разделить на присоединенные, свободные и вычисляемые.

Присоединенные элементы – присоединяются к полю таблицы. При вводе значения в присоединенный элемент Access автоматически корректирует поле таблицы в текущей записи.

Свободные элементы – это элементы, сохраняющие введенное значение в самой форме, но не корректирующие содержимое полей таблиц.

Вычисляемые элементы основываются на выражениях, функциях или вычислениях. Они так же, как и свободные, не корректируют табличных полей. Выражение в вычисляемом поле должно начинаться со знака равно.

Виды элементов управления:

Надпись – элемент управления, предназначенный для вывода описательного текста, например, заголовков, подписей или пояснений в форме или отчете. Для надписей можно использовать различные размеры, стили и начертание шрифта.

Поле – это элемент, в котором пользователь вводит или редактирует данные. Поля могут быть присоединенными или свободными. Поля можно использовать из таблиц или запросов, или они могут включать рассчитываемые выражения.

Выключатели , *переключатели* *и флажки* – элементы, используемые с данными логического типа. Каждый из них может применяться отдельно, чтобы показать одно из двух возможных состояний: «да» или «нет», «включено» или «отключено», «истина» или «ложь».

Группа – элемент, который может содержать выключатели, переключатели и флажки. При помещении этих элементов в поле группы они работают в комплексе, а не по отдельности. В каждый момент времени в группе может быть выбран только один элемент. Максимальное количество элементов в группе – четыре. Для включения большего количества кнопок необходимо создать раскрывающийся список. Обычно группа присоединена к отдельному полю. Каждая кнопка в группе возвращает свое исходное состояние, а группа в свою очередь передает единственную возможность выбора (номер выбранного элемента) присоединенному полю. Сами кнопки присоединены только к группе.

Список – элемент, который выдает на экран постоянно открытый список данных. Чтобы выбрать элемент в списке, следует щелкнуть на нем мышью или поместить на него указатель и нажать клавишу «Enter». Значение выбранного элемента передается присоединенному полю.

Поле со списком – элемент, который в отличие от списка дает возможность вводить значение, которого в списке нет. Первоначально поле со списком выводится на экран как отдельное поле со стрелкой, позволяющей открыть список.

Кнопка – используется для запуска макрокоманд и процедур VBA.

Рисунок – используется для вывода в форме неизменного рисунка.

Свободная рамка объекта – позволяет создать рамку для свободного объекта OLE, который остается неизменным при переходе от записи к записи.

Присоединенная рамка объекта – предназначена для объектов, сохраненных в поле источника записей. Для разных записей будут разные объекты.

Разрыв страницы – используется для начала нового экрана в форме или новой страницы в печатной форме.

Вкладка – может рассматриваться как контейнер для других элементов управления. Каждая определенная вкладка представляет собой отдельную страницу области вкладок, на которую можно поместить другие элементы.

Подчиненная форма – форма, встроенная в другую форму.

Линия и *прямоугольник* – графические элементы для придания выразительности форме.

Пиктограмма *Другие элементы* позволяет раскрыть список других элементов управления, перечень которых зависит от установленного на данном компьютере программного обеспечения.

Создать элемент управления формы можно одним из следующих способов:

- перетащить поле из окна списка полей, открываемого командой ***Вид/Список полей***, на требуемую область формы. При этом элемент автоматически присоединяется к перетаскиваемому полю, а текст подписи создается в качестве надписи из имени поля;
- щелкнуть на соответствующей кнопке панели элементов (рисунок 7) и затем добавить данный элемент на форму. При этом если предварительно была нажата пиктограмма , для большинства элементов управления будет запущен мастер, помогающий создавать данный элемент.

Рисунок 7 – Панель инструментов

Выделение, изменение размеров и перетаскивание элементов управления осуществляются аналогично работе с объектами Windows. Внешний вид элемента управления может быть изменен путем изменения его свойств.

Формат ссылки на элемент управления формы приведен на с. 23.

1.2 Отчеты

Отчет – это гибкое средство для представления данных при выводе на печать.

Отчеты могут быть открыты в трех режимах: в режиме конструктора, режиме предварительного просмотра и режиме просмотра образца. Режим конструктора используют для создания или модифицирования отчета. Режим предварительного просмотра позволяет просматривать данные в том виде, в котором они будут размещаться на печатных страницах. В режиме просмотра образца выводятся основные элементы макета отчета с образцами данных, демонстрирующих представление данных в отчете. Режим открытия определяется с помощью пунктов меню ***Вид***.

Отчет создается на базе таблицы или запроса. Так же как и форма, отчет может включать элементы управления, рассмотренные выше.

Типы отчетов

Отчет в один столбец. Представляет собой один столбец текста, содержащий значения всех полей каждой записи. Имеется возможность изменения макета отчета в режиме конструктора и расположения полей не в один столбец.

Ленточный отчет. В этом случае для каждого поля записи используется свой столбец, а значения всех полей каждой записи выводятся по строкам, причем каждое в своем столбце. Заголовки печатаются сверху на каждой странице.

Итоговые отчеты (с группировкой). Наиболее распространенный тип. В таких отчетах данные объединяются в группы, а в конце отчета или групп рассчитываются итоговые значения.

Почтовые наклейки. Специализированный вид многоколоночного отчета с группировкой по именам и адресам.

Вся информация в отчете разбивается на разделы (заголовок, верхний колонтитул, область данных, нижний колонтитул, примечание), каждый из которых имеет специальное назначение. При печати разделы располагаются на страницах в указанном порядке.

Для добавления или удаления областей заголовка, примечаний и колонтитулов отчета необходимо выполнить команду ***Вид/Заголовок/примечание отчета*** или команду ***Вид/Колонтитулы***. Области заголовка и примечаний, а также верхнего и нижнего колонтитулов всегда добавляются парами. Если не требуется выводить на печать одну из областей такой пары, следует задать для лишней области значение **Нет** для свойства **Вывод на экран** или же удалить все элементы управления из этой области и установить для нее нулевую высоту или задать значение 0 для свойства **Высота**.

Пользователь имеет возможность разработать отчет самостоятельно в режиме конструктора или создать отчет с помощью мастера.

При разработке отчета в режиме конструктора в отчете могут использоваться элементы управления, аналогичные элементам управления форм: (надписи, поля, флажки или переключатели, группы, содержащие флажки или переключатели, линии, прямоугольники, подчиненные отчеты, рамки для рисунка или другого объекта, диаграммы). Элементы управления могут быть свободными и присоединенными. Свободные элементы могут

использоваться для вычислений. Выражение в вычисляемом поле должно начинаться со знака равно.

Ссылка на элементы управления в отчетах:

Reports!ИмяОтчета!ИмяЭлементаУправления.Свойство

На элементы управления текущего отчета можно ссылаться просто по именам.

Группировка записей в отчете

В отчетах допускается группировка не более чем по 10 полям или выражениям. Для создания группировки необходимо открыть отчет в режиме конструктора и выполнить команду ***Вид/Сортировка и группировка***. В открывшемся окне (рисунок 8) указывается порядок сортировки данных в отчете.

Рисунок 8 – Окно Сортировка и группировка

Чтобы создать уровень группировки по определенному полю и определить другие свойства группы, необходимо задать значение **Да** по крайней мере для одного из свойств **Заголовок группы** или **Примечание группы**. Назначение остальных свойств следующее:

- **Группировка** – 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** – указывает величину интервалов, на которые должны быть разбиты значения в группируемых записях. Допустимые интервалы группировки определяются типом данных и значением свойства **Группировка** поля или выражения, по которому проводится группировка. Например, значение 1, используемое по умолчанию, может определять группировку записей по первому символу в названии товара. Чтобы группировать для текстового типа поля данные по первым трем символам, необходимо для данного свойства указать значение 3.
- **Не разрывать** – указывает, следует ли печатать разные элементы данной группы на одной странице отчета (**Нет** – группа может печататься на разных страницах; **Полную группу** – вся группа печатается на одной странице; **Первую область данных** – заголовок группы печатается на текущей

странице только в том случае, если вместе с ним помещается первая запись области данных).

Для расчета итогового значения по группе записей необходимо добавить вычисляемое поле в заголовок группы или в примечание группы. Для расчета итогового значения по всем записям отчета необходимо добавить вычисляемое поле в заголовок отчета или в примечание отчета. При вычислениях можно использовать статистические функции.

2 Порядок выполнения работы

1 Изучить виды форм и отчетов, последовательность их создания, их свойства, элементы управления и их свойства.

2 Получить задание у преподавателя.

3 С помощью Access в соответствии с полученным заданием создать требуемые формы и отчеты.

4 Оформить отчет по лабораторной работе.

3 Содержание отчета

1 Название работы и цель работы.

2 Исходное задание.

3 Макеты форм и отчетов.

4 Выводы.

4 Контрольные вопросы

1 Что такое подчиненная форма и каким образом устанавливается связь данных подчиненной формы с главной формой?

2 Чем отличаются присоединенные элементы управления от свободных?

3 Каким образом в форме можно представить разные графические изображения для разных записей таблицы?

4 Как создать вычисляемое поле в форме или отчете?

5 Для чего используется группировка данных в отчете?

6 Как защитить данные от изменения при работе с данными таблицы через форму?

7 Как организовать итоговые вычисления в отчете?

8 Как вывести данные в списке в несколько колонок?

9 Как скрыть от отображения присоединенный столбец в поле со списком?

Литература

1 Система помощи Microsoft Access 2003.

2 Кузнецов, А. Microsoft Access 2003. Русская версия / А. Кузнецов. – СПб. : Питер, 2006.

ЛАБОРАТОРНАЯ РАБОТА №4

Автоматизация приложения с помощью модулей

Цель работы: Научиться создавать с помощью Access модули для автоматизации работы приложения.

1 Теоретические сведения

1.1 Общие сведения об автоматизации приложения

Автоматизировать работу приложения базы данных можно с помощью модулей и макросов. Модули являются более универсальным средством, чем макросы, и рекомендуются Microsoft для автоматизации при создании новых приложений. Перечислим вопросы автоматизации приложения, которые нельзя решить с помощью макросов или целесообразно решать с помощью модулей:

- Обработка ошибок в приложении.
- Создание пользовательских функций.
- Создание новых объектов (таблиц, форм, отчетов, запросов) во время работы приложения.
- Использование системных функций Windows.
- Необходимость обработки отдельных записей.

Модуль – это набор объявлений и процедур на языке VBA, собранных в одну программную единицу. Существует два основных типа модулей: стандартные модули и модули класса.

В стандартных модулях содержатся общие процедуры и функции, не связанные ни с каким объектом, а также часто используемые процедуры. Список стандартных модулей базы данных отображается на вкладке **Модули** в окне базы данных. Новый стандартный модуль создается аналогично остальным объектам Access (формам, отчетам и т.д.) путем выбора кнопки **Создать** в окне базы данных при выбранных объектах **Модули**.

Модули форм и модули отчетов являются модулями класса, связанными с определенной формой или отчетом. Они часто содержат процедуры обработки событий, запускаемых в ответ на событие в форме или отчете. Процедуры обработки событий используются для управления поведением формы или отчета и их откликом на события, например такие, как нажатие кнопки. При создании первой процедуры обработки события для формы или отчета автоматически создается связанный с ней модуль формы или отчета. В процедурах модулей форм и отчетов могут содержаться вызовы процедур, добавленных в стандартные модули.

Модуль класса создается путем щелчка по кнопке построителя для одного из свойств событий формы, отчета или элемента управления с последующим выбором пункта меню ***Программа***. Для открытия модуля класса формы необходимо в режиме конструктора щелкнуть по кнопке  главной кнопочной панели.

Каждый модуль состоит из раздела описаний (Declaration Section) и одной или нескольких процедур или функций. В разделе описаний модуля содержатся описания используемых в модуле констант и переменных.

1.2 События в формах и отчетах

Событие – это определенное действие, которое происходит над или возникает в определенном объекте. Обычно события возникают вследствие действий пользователя.

С помощью процедур обработки события или макроса возможно определение собственных реакций на события, происходящие в форме, отчете или элементе управления. Чтобы в ответ на событие запустить модуль, следует открыть окно свойств для формы, отчета или элемента управления, найти соответствующее событию свойство и установить в качестве его значения вызов соответствующей процедуры.

Рассмотрим наиболее часто используемые события в формах и элементах управления. При описании событий в скобках будут перичисляться объекты, для которых возможно это событие.

События данных возникают при вводе, удалении или изменении данных в форме или элементе управления, а также при перемещении фокуса с одной записи на другую.

После подтверждения Del (AfterDelConfirm) (формы). Возникает после подтверждения пользователем удаления записей и фактического удаления записей или после отмены удаления.

После вставки (AfterInsert) (формы). Возникает после добавления новой записи в базу данных.

После обновления (AfterUpdate) (формы, элементы управления). Возникает после обновления данных в элементе управления или записи при потере фокуса элементом управления или записью. Событие возникает для новых и существующих записей.

До подтверждения Del (BeforeDelConfirm) (формы). Возникает после удаления одной или нескольких записей, но до открытия диалогового окна Microsoft Access с приглашением подтвердить или отменить удаление. Данное событие возникает после события *Удаление (Delete)*.

До вставки (BeforeInsert) (формы). Возникает при вводе первого символа в новую запись, но до добавления записи в базу данных.

До обновления (BeforeUpdate) (формы, элементы управления). Возникает при потере фокуса элементом. Событие возникает для новых и существующих записей.

Изменение (Change) (элементы управления). Возникает при изменении содержимого в поле или в поле со списком.

Удаление (Delete) (формы). Возникает при удалении записи пользователем, но до подтверждения удаления и до фактического удаления записи.

Отсутствие в списке (NotInList). Возникает при вводе в поле со списком значения, отсутствующего в списке поля со списком.

При обновлении (Updated) (элементы управления). Возникает при изменении данных в объекте.

События клавиатуры возникают при вводе с клавиатуры, а также при передаче нажатий клавиш с помощью инструкции SendKeys.

Клавиша вниз (KeyDown) (формы, элементы управления). Возникает при нажатии пользователем любой клавиши на клавиатуре в тот момент, когда элемент управления или форма имеют фокус. Форма может получить фокус только в том случае, когда все ее видимые элементы управления недоступны или когда в форме нет элементов управления. Если пользователь нажимает и удерживает клавишу, то событие *Клавиша вниз (KeyDown)* возникает многократно.

Нажатие клавиши (KeyPress) (формы, элементы управления). Возникает, если пользователь нажимает и отпускает клавишу в момент, когда элемент управления или форма имеют фокус. Если пользователь нажимает и удерживает клавишу, то событие *Нажатие клавиши* возникает многократно.

Клавиша вверх (KeyUp) (формы, элементы управления). Возникает, если пользователь отпускает нажатую клавишу в момент, когда элемент управления или форма имеют фокус. Если пользователь нажимает и удерживает клавишу, то событие *Клавиша вверх* возникает после всех событий *Клавиша вниз* и *Нажатие клавиши*.

События мыши возникают при действиях с мышью.

Нажатие кнопки (Click) (формы, элементы управления). Для элемента управления это событие возникает, когда пользователь нажимает и быстро отпускает левую кнопку мыши при указателе, установленном на элементе управления. Для формы это событие возникает, когда пользователь выбирает при помощи мыши область выделения записи, а также область формы вне раздела или элемента управления.

Двойное нажатие кнопки (DblClick) (формы, элементы управления). Для элемента управления это событие возникает, когда пользователь дважды быстро нажимает и отпускает левую кнопку мыши при указателе, установленном на элементе управления или присоединенной к нему надписи. Для формы это событие возникает, когда пользователь дважды быстро нажимает и отпускает кнопку мыши при указателе, установленном в области выделения записи или в пустой области формы.

Кнопка вниз (MouseDown) (формы, элементы управления). Возникает в момент нажатия кнопки мыши при указателе, установленном на форме или на элементе управления.

Перемещение указателя (MouseMove) (формы, элементы управления). Возникает при перемещении указателя по форме или элементу управления.

Кнопка вверх (MouseUp) (формы, элементы управления). Возникает в момент отпускания нажатой кнопки мыши при указателе, установленном на форме или на элементе управления.

События фокуса возникают, когда форма или элемент управления теряют или получают фокус, а также в момент, когда форма или отчет становятся активным или неактивным окном.

Включение (Activate) (формы, отчеты). Возникает, когда окно формы или отчета становится активным.

Отключение (Deactivate) (формы, отчеты). Возникает при выполнении действия, в результате которого активным должно стать другое окно Microsoft Access, но до того, как это окно действительно станет активным.

Вход (Enter) (элементы управления). Возникает перед фактическим получением фокуса элементом управления от другого элемента управления в той же форме или при открытии формы. Данное событие возникает до события *Получение фокуса (GotFocus)*.

Выход (Exit) (элементы управления). Возникает непосредственно перед переводом фокуса на другой элемент управления в той же форме. Данное событие возникает до события *Потеря фокуса (LostFocus)*.

Получение фокуса (GotFocus) (формы, элементы управления). Возникает при получении фокуса элементом управления или формой, не имеющей активных или доступных элементов управления. Форма может получить фокус только в том случае, когда все ее видимые элементы управления недоступны или когда в форме нет элементов управления.

Потеря фокуса (LostFocus) (формы, элементы управления). Возникает при потере фокуса элементом управления или формой. Форма может получить фокус только в том случае, когда все ее видимые элементы управления недоступны или когда в форме нет элементов управления.

События окна возникают при открытии, изменении размеров или закрытии формы или отчета.

Закрытие (Close) (формы, отчеты). Возникает при закрытии формы или отчета.

Загрузка (Load) (формы). Возникает при открытии формы и выводе в ней записей. Данное событие возникает после события *Открытие (Open)*.

Открытие (Open) (формы, отчеты). Возникает при открытии формы, но до вывода первой записи. При открытии отчета – до начала печати отчета.

Изменение размера (Resize) (формы). Возникает при изменении размеров и первом открытии формы.

Выгрузка (Unload) (формы). Возникает при закрытии формы и выгрузке ее записей, но до удаления формы с экрана. Данное событие возникает до события *Закрытие (Close)*.

При обработке ошибок можно использовать событие *Ошибка (Error)* (формы, отчеты). Возникает при возникновении ошибки выполнения Microsoft Access, когда фокус находится в форме или отчете. В число таких ошибок включаются ошибки ядра базы данных Microsoft Jet, но не включаются ошибки выполнения Visual Basic.

1.3 Краткие сведения об объектах приложения

Microsoft Access включает следующие основные подсистемы: ядро приложения Application Engine, обеспечивающее пользовательский интерфейс приложения и ядро базы данных Microsoft Jet (Jet DBEngine), обеспечивающее управление хранением и доступом к данным. Каждая система имеет свою

иерархию объектов. Рассмотрим наиболее часто используемые объекты и семейства объектов (коллекции) ядра приложения.

Семейство Forms содержит все формы, открытые в данный момент в базе данных Microsoft Access. Семейство Forms используется в программах VBA или в выражениях для ссылок на формы, открытые в данный момент.

Объект Form представляет конкретную форму Microsoft Access и является компонентом семейства Forms, в которое входят все текущие открытые формы. В рамках семейства Forms отдельные формы нумеруются с помощью индекса, начинающегося с нуля. Допускаются ссылки на объект Form в семействе Forms либо по имени соответствующей формы, либо по ее индексу в семействе. Например *Forms![Микросхемы]*, *Forms("Микросхемы")*, *Forms(0)*. При ссылках на определенную форму рекомендуется применять ссылки по имени, поскольку индекс формы в семействе может измениться. Имена, содержащие пробелы и другие специальные символы, следует заключать в квадратные скобки.

Каждый объект Form содержит семейство Controls, включающее все элементы управления в форме (объекты Control). При ссылках на конкретные элементы управления в форме допускаются как явные, так и неявные ссылки на семейство Controls. Программа с неявными ссылками выполняется быстрее. Пример неявной ссылки: Forms!Микросхемы!Обозначение. Пример явной ссылки: Forms!Микросхемы.Controls!Обозначение. При ссылке в модуле формы на элементы управления этой же формы лучше (по соображениям скорости выполнения) использовать синтаксис вида *Me!ИмяЭлементаУправления*.

Для работы с объектами используются их методы и свойства. Рассмотрим несколько наиболее часто используемых методов для работы с формами и элементами управления.

Метод Requery обновляет данные, выводящиеся в указанной форме или в элементе управления в активной форме с помощью выполнения повторного запроса к источнику данных формы или элемента управления. Синтаксис:

имяОбъекта.***Requery***

Аргумент *имяОбъекта* представляет имя объекта Form или Control, определяющий форму или элемент управления, который необходимо изменить.

Метод ***Requery*** отображает результаты добавления или удаления записей, выполненного после последнего запроса к источнику записей. Метод ***Requery*** выполняется быстрее, чем макрокоманда *Обновление (Requery)*.

Метод SetFocus переводит фокус на указанную форму или на элемент управления активной формы, а также на поле активного объекта в режиме таблицы. Синтаксис

имяОбъекта.***SetFocus***

Аргумент имеет тот же смысл, что и в методе ***Requery***.

Метод ***SetFocus*** применяют при необходимости перевести фокус на определенное поле или элемент управления для того, чтобы все вводящиеся

пользователем данные были адресованы этому объекту.

Для считывания значений некоторых свойств элемента управления необходимо, чтобы этот элемент управления имел фокус. Например, для того чтобы считать значение свойства Text поля, необходимо перевести фокус на это поле.

Допускается перемещение фокуса только на видимый элемент управления или форму. Нельзя поместить фокус на элемент управления, если свойство Доступ (Enabled) этого элемента управления имеет значение False. Однако допускается перевод фокуса на элемент управления, у которого свойство Блокировка (Locked) имеет значение True.

Наиболее просто автоматизация работы приложения обеспечивается за счет использования методов объекта DoCmd [1].

1.4 Объект DoCmd

Методы, определенные для объекта DoCmd, позволяют запускать макрокоманды Microsoft Access из программ Visual Basic. Для использования методов объекта DoCmd следует применять следующий синтаксис:

DoCmd.метод [арг1, арг2, ...]

Аргументы синтаксиса методов объекта DoCmd:

Метод – имя одного из методов, поддерживаемых объектом;

арг1, арг2, ... – аргументы указанного метода.

Большинство из методов, определенных для объекта DoCmd, имеют аргументы, некоторые из которых являются обязательными, а другие необязательными. Если необязательный аргумент опущен, то при выполнении макрокоманды подразумевается значение данного аргумента по умолчанию.

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.

Большинству макрокоманд соответствуют методы DoCmd, имена которых совпадают с английскими именами макрокоманд. Свойства у объекта DoCmd отсутствуют.

Метод Close закрывает объект. Метод имеет следующий синтаксис:

DoCmd.Close [типОбъекта, имяОбъекта], [сохранение]

Аргументы метода:

типОбъекта - одна из следующих встроенных констант: acDefault (значение по умолчанию), acForm (форма), acMacro (макрос), acModule (модуль), acQuery (запрос), acReport (отчет), actable (таблица);

имяОбъекта – строковое выражение, представляющее допустимое имя объекта, тип которого указан в аргументе *типОбъекта*.

сохранение - одна из встроенных констант: acSaveNo (выход без сохранения), acSavePrompt (подтверждение - значение по умолчанию), acSaveYes (сохранение без подтверждения).

Если оставлены пустыми аргументы *типОбъекта* и *имяОбъекта* (по умолчанию аргументу *типОбъекта* присваивается значение acDefault), Microsoft Access закрывает активное окно. Для того чтобы определить аргумент сохранение и оставить аргументы *типОбъекта* и *имяОбъекта* пустыми, необходимо ввести запятые, представляющие аргументы *типОбъекта* и *имяОбъекта*.

Метод ***DeleteObject*** удаляет объект. Метод имеет следующий синтаксис:

DoCmd.DeleteObject *[типОбъекта, имяОбъекта]*

Аргументы метода:

типОбъекта - одна из констант аналогично методу Close;

имяОбъекта - строковое выражение, представляющее допустимое имя объекта, тип которого указан в аргументе *типОбъекта*.

Если оставлены пустыми аргументы *типОбъекта* и *имяОбъекта*, Microsoft Access удаляет объект, выделенный в окне базы данных.

Метод ***FindRecord*** позволяет найти запись в соответствии с заданным образцом. Метод имеет следующий синтаксис:

DoCmd.FindRecord *образец [, совпадение] [, регистр] [, поиск] [, формат] [, текущееПоле] [, сНачала]*

Метод ***FindRecord*** использует следующие аргументы:

образец - выражение, значением которого является текст, число или дата, задающее образец для поиска;

совпадение - одна из следующих констант: ***acAnywhere*** (с любой частью поля), ***acEntire*** (со значением поля) – значение по умолчанию, ***acStart*** (с начальными символами в значении поля);

регистр - значение True (-1) задает поиск с учетом регистра символов, а False (0) - поиск без учета регистра. Если оставить данный аргумент пустым, подразумевается значение по умолчанию (False).

поиск - одна из следующих встроенных констант: acDown (вниз), acSearchAll (везде) - значение по умолчанию, acUp (вверх);

формат - значение True задает поиск с учетом формата полей, а значение False - поиск данных в том виде, в котором они сохраняются в базе данных. Значение по умолчанию - False.

текущееПоле - одна из следующих встроенных констант: acAll (все поля), acCurrent (текущее поле) – значение по умолчанию;

сНачала - значение True задает начало поиска с первой записи, False задает начало поиска с текущей записи. Значение по умолчанию - True.

Следующая конструкция обнаружит в текущем поле фамилию «Рост». «Ростов» или «рост» найдены не будут.

DoCmd.FindRecord *"Рост", True,,, True*

Метод FindNext позволяет найти следующую запись, удовлетворяющую аргументам предыдущего метода ***FindRecord.*** Синтаксис:

DoCmd.FindNext

Данный метод не требует аргументов.

Метод GoToRecord позволяет перейти на определенную запись. Синтаксис:

DoCmd.GoToRecord [типОбъекта, имяОбъекта] [, запись] [, смещение]

Метод GoToRecord использует следующие аргументы:

типОбъекта - одна из следующих встроенных констант: acActiveDataObject (активный объект, значение по умолчанию), acDataForm (форма), acDataQuery (запрос), acDataTable (таблица);

имяОбъекта - строковое выражение, представляющее допустимое имя объекта, тип которого указан в аргументе *типОбъекта*.

запись - одна из следующих встроенных констант: acFirst (первая), acGoTo (с указанным номером), acLast (последняя), acNewRec (новая), acNext (следующая) - значение по умолчанию, acPrevious (предыдущая);

смещение - числовое выражение, представляющее число записей, на которое отстоит искомая запись от текущей вперед или назад, если в аргументе *запись* заданы константы acNext или acPrevious, или номер искомой записи, если в аргументе запись задана константа acGoTo. Значение выражения должно представлять допустимый номер записи.

Если оставлены пустыми аргументы *типОбъекта* и *имяОбъекта*, то подразумевается активный объект.

Пример перехода к 7 записи:

DoCmd.GoToRecord acDataForm, "Сотрудники", acGoTo, 7

Метод OpenForm открывает заданную форму. Метод имеет следующий синтаксис:

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

Метод *OpenForm* использует следующие аргументы:

имяФормы - строковое выражение, представляющее допустимое имя формы в текущей базе данных;

режим - одна из следующих встроенных констант: acDesign (конструктор), acFormDS (таблица), acNormal (открытие формы в режиме формы) - значение по умолчанию, acPreview (просмотр);

имяФайла - 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) - значение по умолчанию, acFormReadOnly (только чтение);

режимОкна - одна из следующих встроенных констант: acDialog (окно диалога), acHidden (невидимое), acIcon (значок), acWindowNormal (обычное) - значение по умолчанию;

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.

Предположим, например, что в режиме ленточной формы открывается список авторов. Для того чтобы при открытии формы установить фокус на запись об определенном авторе, достаточно указать фамилию автора в аргументе *аргументыОткрытия*, а затем вызвать метод FindRecord, чтобы переместить фокус на запись для автора с нужной фамилией.

Максимальная длина строки в аргументе условиеWhere составляет 32 768 символов.

В данном 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'"
```

Метод ***OpenQuery*** позволяет открыть запрос. Метод имеет следующий синтаксис:

DoCmd.OpenQuery *имяЗапроса [, режим] [, режимДанных]*

Метод ***OpenQuery*** использует следующие аргументы:

имяЗапроса - строковое выражение, представляющее допустимое имя запроса в текущей базе данных.

режим - одна из следующих встроенных констант: acViewDesign (конструктор), acViewNormal (таблица, значение по умолчанию), acViewPreview (просмотр);

режимДанных - одна из следующих встроенных констант: acAdd (добавление), acEdit (изменение) - значение по умолчанию, acReadOnly (только чтение).

Метод ***OpenReport*** позволяет открыть отчет. Метод имеет следующий синтаксис:

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

Метод ***OpenReport*** использует следующие аргументы:

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);

имяФайла - строковое выражение, представляющее допустимое имя запроса в текущей базе данных,

условиеWhere - строковое выражение, представляющее допустимое предложение SQL WHERE без ключевого слова WHERE.

Метод ***OpenTable*** позволяет открыть таблицу. Метод имеет синтаксис:

DoCmd.OpenTable *имяТаблицы [, режим] [, режимДанных]*

Метод ***OpenTable*** использует следующие аргументы:

имяТаблицы - строковое выражение, представляющее допустимое имя таблицы в текущей базе данных;

режим - одна из следующих встроенных констант: acViewDesign (конструктор), acViewNormal (таблица - значение по умолчанию), acViewPreview (просмотр);

режимДанных - одна из следующих встроенных констант: acAdd (добавление), acEdit (изменение, значение по умолчанию), acReadOnly (только чтение).

Метод ***RunCommand*** выполняет команду встроенного меню или встроенной панели инструментов. Синтаксис:

DoCmd.RunCommand *команда*

Аргумент *команда* является встроенной константой, указывающей, какая именно команда встроенного меню или панели инструментов выполняется. Каждой команде меню или панели инструментов Microsoft Access соответствует константа, которая используется в методе ***RunCommand*** для выполнения этой команды в программе VBA. Перечень всех констант для аргумента команда содержится в разделе «Константы» метода ***RunCommand***. Не допускается применение метода RunCommand для выполнения команды специального меню или специальной панели инструментов. Он используется только для встроенных меню или панелей инструментов.

Метод ***SetWarnings*** позволяет запретить или разрешить вывод сообщений Access. Синтаксис:

DoCmd.SetWarnings *режим*

Значение True (-1) аргумента *режим* задает включение режима вывода системных сообщений, значение False (0) отключает вывод сообщений. Режим вывода сообщений, отключенный в программе Visual Basic, останется отключенным даже после прерывания программы нажатием клавиш CTRL+BREAK или после прерывания выполнения программы на точке останова VBA.

2 Порядок выполнения работы

1 Изучить принципы автоматизации приложения, объекты приложения, методы объекта DoCmd. Для этого использовать систему помощи Access и лабораторный практикум.

2 Получить задание у преподавателя.

3 С помощью Access в соответствии с полученным заданием создать модули для реализации требуемых процедур управления работой приложения.

4 Оформить отчет по лабораторной работе.

3 Содержание отчета

1 Название работы и цель работы.
2 Исходное задание.
3 Исходный текст модулей.
4 Выводы.

4 Контрольные вопросы

1 Чем отличается модуль класса от стандартного модуля?
2 Для чего используются события в формах и отчетах?
3 С помощью какого метода можно выполнить повторный запрос к источнику данных списка или поля со списком?
4 Как в модуле сослаться на поле формы или отчета?
5 Какие объекты принадлежат к семейству Forms?
6 Как с помощью модуля можно запустить на исполнение запрос?
7 Как с помощью модуля перейти к требуемой записи в форме?
8 Какие задачи работы с приложением можно решить, используя модули?
9 Как в модуле можно получить значение поля формы?

Литература

1 Дженнингс, Р. Microsoft Access 97 в подлиннике / Р. Дженнингс. – СПб. : BHV – Санкт-Петербург, 1997.
2 Система помощи Microsoft Access 2003.
3 Баркер, С.Ф. Профессиональное программирование в Microsoft Access 2002 / С. Ф. Баркер. – М. : Вильямс, 2002.

Св. план 2008, поз. 19

Учебное издание

Станкевич Андрей Владимирович

СУБД Access

Лабораторный практикум
по курсу «Прикладные пакеты САПР проблемно-ориентированных электронных вычислительных средств»
для студентов специальности I-40 02 02
«Электронные вычислительные средства»
дневной формы обучения

Редактор Т. П.Андрейченко
Корректор Е. Н. Батурчик

Подписано в печать	Формат 60×84 1/16.	Бумага офсетная.
Гарнитура «Таймс».	Печать ризографическая.	Усл. печ. л.
Уч.-изд. л. 3,0.	Тираж 100 экз.	Заказ 164.

Издатель и полиграфическое исполнение: Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
ЛИ №02330/0056964 от 01.04.2004. ЛП №02330/0131666 от 30.04.2004.
220013, Минск, П. Бровки, 6