2 ГЛАВА Запросы к базе данных

В данной главе рассматриваются вопросы поиска информации в базах данных MS Access. В разделе 2.1 материал излагается в соответствии с европейским стандартом ECDL (European Computer Driving Licence), базовый уровень (пятая версия программы (syllabus), взята с сайта http://www.ecdl.com/programmes/, см. приложение 2, содержащее раздел 5 (базы данных) программы ECDL). В разделах 2.2 и 2.3 рассматриваются дополнения, знание которых требуется в соответствии с расширенным уровнем ECDL (advanced level, версия вторая, см. приложение 3, раздел 5.3 стандарта ECDL). В целом материал в указанных разделах излагается в соответствии с приведенными документами, но вопросы поиска информации непосредственно в таблицах (5.4.1) рассматриваются в предыдущей главе.

В разделе 2.3.5 рассматриваются примеры использования рекурсивного и внешнего соединений таблиц в запросах. В соответствии с программой ECDL (расширенный уровень), этот материал относится к разделу, в котором излагаются сведения о разработке таблиц (см. приложение 3, разделы 5.2.2.6, 5.2.2.8), и поэтому теоретически рассматривается в разделе 1.4.4.

В разделе 2.3.6 излагается материал, не вошедший в состав программ ECDL, но представляющий, как нам кажется, определенный интерес при практической разработке баз данных. Раздел 2.3.7 отражает особенности работы в Access 2010.

2.1. Поиск данных

В данном разделе рассматриваются вопросы, связанные с извлечением информации из таблиц базы данных:

- создание простого запроса;
- использование нескольких таблиц в запросе;

- использование критериев отбора в запросе;
- применение сортировки данных в запросе.

Поиск данных может осуществляться как непосредственно в таблицах (в режиме Таблица), так и с помощью специального объекта Запросы. Первый вариант позволяет искать и отбирать данные в соответствии с относительно простыми условиями поиска, второй является более развитым инструментом и позволяет осуществлять поиск по разнообразным сложным критериям.

2.1.1. Простой запрос

Как отмечалось ранее, простой запрос можно составить непосредственно из расширенного фильтра. Для этого с помощью навигационной панели откроем таблицу Data и перейдем на вкладку Главная. Далее щелкнем на кнопке Расширенный фильтр в области Сортировка и фильтр (рис. 2.1).

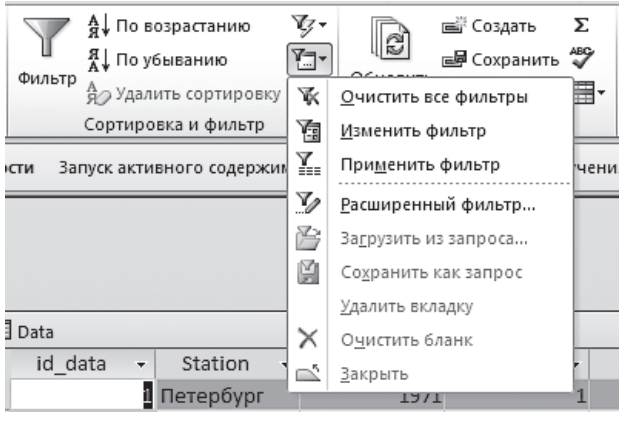

Рис. 2.1. Переход к расширенной фильтрации

На экране появится окно конструктора запросов с заголовком ДанныеФильтр1. Это окно состоит из двух областей, расположенных одна под другой. В верхней половине отображается набор таблиц, участвующих в запросе, в нижней половине можно сконструировать поля для вывода. При необходимости границу между областями можно переместить.

Сконструировать динамическую таблицу для вывода данных в соответствии с выбранным критерием отбора проще всего перетаскиванием полей из верхней половины конструктора в соответствующие поля таблицы внизу. Попробуем составить запрос на выборку данных по метеостанции «Сортавала» из таблицы Data. Надо решить заранее, какую информацию мы хотим увидеть в результате выполнения запроса. Например, такую: станция, год, месяц, средняя температура. По очереди перетащим эти поля в столбцы таблицы. Так как код станции «Сортавала» — 3, запишем его в строку критериев столбца id_station. Результат составления расширенного фильтра представлен на рис. 2.2.

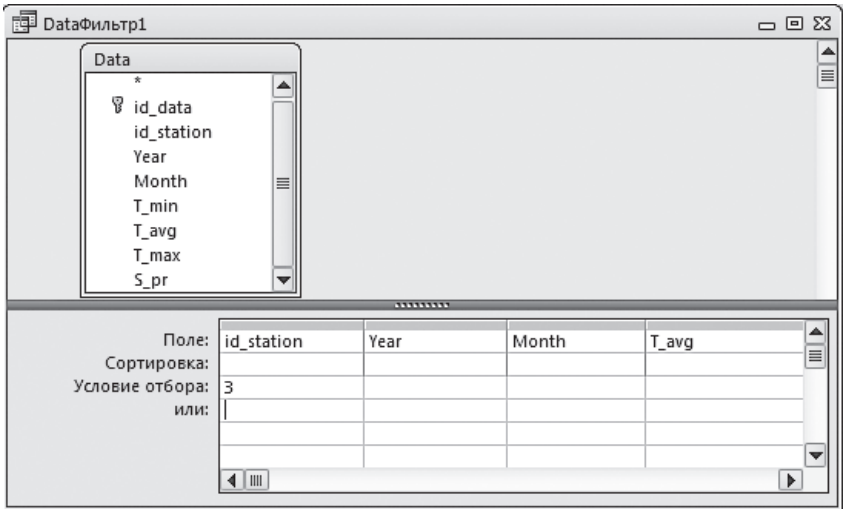

Рис. 2.2. Расширенный фильтр

Для того чтобы запустить фильтрацию, следует вновь щелкнуть на кнопке Применить фильтр. Сохранив таблицу, мы получаем возможность в дальнейшем использовать фильтрацию по заданному правилу, щелкнув на кнопке с изображением воронки (Применить фильтр). Однако если есть необходимость фильтровать данные в таблице по нескольким разным критериям, фильтр необходимо сохранить как запрос. Для этого в режиме фильтрации следует щелкнуть все на той же кнопке расширенной фильтрации и выбрать позицию Сохранить как запрос (рис. 2.3).

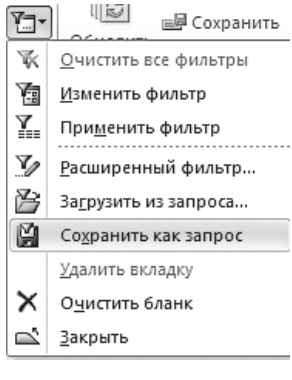

Рис. 2.3. Сохранение фильтра как запроса

Отметим, что в целом инструмент расширенной фильтрации имеет скорее учебное, чем практическое назначение. Обычно намного удобнее пользоваться непосредственно запросами.

В практической работе приходится использовать в одном запросе сразу несколько таблиц, выбор которых обусловлен структурой таблиц БД. Поясним сказанное на

простом примере. Понятно, что использование цифрового кода вместо названия станции в приведенном чуть раньше запросе создает известные неудобства. Нужно помнить коды всех станций, а в реальной базе данных их будет гораздо больше, чем в нашем учебном примере. Текстовые названия станций хранятся в таблице Stations, значит, для создания запроса, в котором можно использовать названия станций, в него нужно включить эту таблицу.

Это можно сделать двумя путями — с помощью конструктора или с помощью мастера. Перейдя на вкладку запросов с помощью навигационной панели и активировав вкладку ленты Создание, в разделе Other справа мы увидим две соответствующие кнопки (рис. 2.4).

Так как более общим является первый способ, вначале рассмотрим его.

Щелкнув по кнопке Конструктор запросов, вызовем окно Добавление таблицы. Так как в общем случае использовать коды в качестве критериев отбора неудобно, добавим в запрос две таблицы (рис. 2.5), благодаря чему получим возможность ввести в качестве критерия отбора название станции вместо ее кода (в режиме фильтрации это, естественно, невозможно).

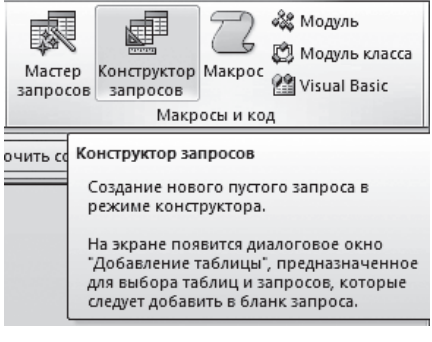

Рис. 2.4. Кнопки создания запросов

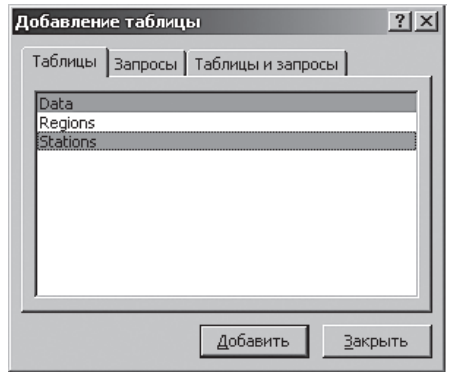

Рис. 2.5. Окно добавления таблицы

Щелкнув на кнопке Добавить, а затем Закрыть, откроем конструктор запросов. Это окно похоже на расширенный фильтр, но есть принципиальные отличия. В частности, верхняя половина конструктора напоминает схему данных, с той разницей, что в ней отражены не все таблицы, а только те, которые мы выбрали для составления запроса. В общем случае все эти таблицы должны быть визуально связаны, иначе запрос не будет работать корректно. Иногда надо проверить связи и удалить лишние, ориентируясь на схему данных, так как конструктор запросов автоматически прорисовывает связи между одноименными полями разных таблиц. Поэтому желательно присваивать всем полям таблиц уникальные названия, за исключением тех полей, по которым действительно предполагается связывать таблицы.

Связи в конструкторе запросов автоматически прорисовываются в соответствии со схемой данных.

В нижней половине конструктора запросов нужно задать сначала имя таблицы (вторая строка), потом имя поля (первая строка). Можно и просто перетащить нужные поля из верхней половины конструктора в нужную ячейку. Ниже при необходимости записываются условия отбора. Так как в таблице Station содержатся названия станций, условием отбора является название станции, которое записано в поле station. Сформированный в окне конструктора запрос показан на рис. 2.6.

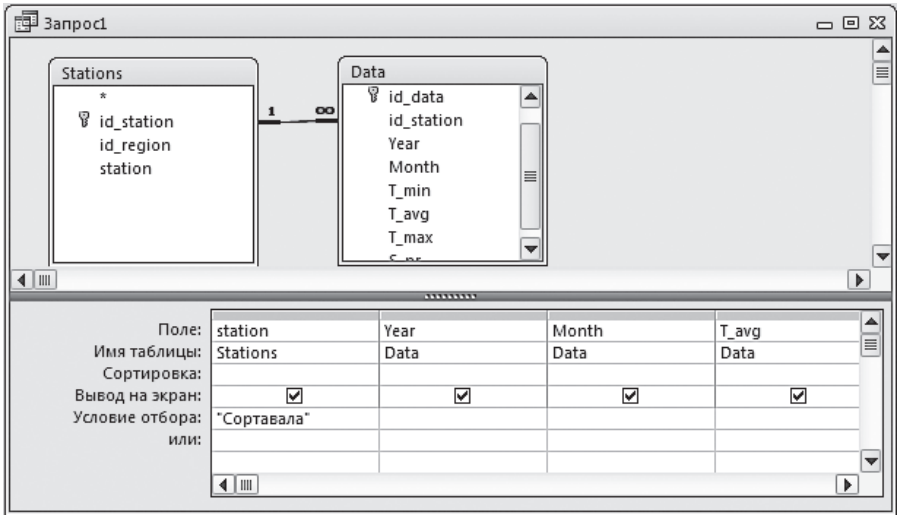

Рис. 2.6. Простой запрос

Щелкнув на кнопке с изображением красного восклицательного знака и подписью Выполнить в левой части меню, запустим запрос на выполнение. Отметим, что левее кнопки Выполнить есть кнопка Режим (таблицы). Щелчок на этой кнопке приводит к похожему результату; но при первом запуске запроса лучше использовать кнопку Выполнить.

Отметим, что создавали запрос мы с помощью вкладки ленты Создание, но для того чтобы редактировать запрос, например, после первого запуска, следует перейти к вкладке Конструктор, что программа и делает автоматически после добавления таблиц в окно конструктора.

При попытке закрыть окно конструктора мы получим сообщение о том, что можно сохранить запрос и присвоить ему имя (рис. 2.7).

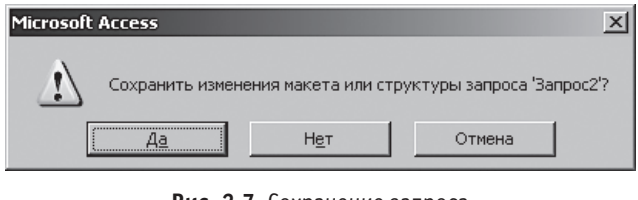

Рис. 2.7. Сохранение запроса

Если в дальнейшем предполагается использовать этот запрос, его следует сохранить при закрытии и в дальнейшем запускать прямо из вкладки запросов навигационной панели.

Сохраним запрос под именем Сортавала.

2.1.2. Модификация запроса

Немного усложним задачу. Предположим, необходимо отобрать данные по двум метеостанциям («Сортавала» и «Петрозаводск»), но так, чтобы средняя температура за месяц была в интервале 10–15 °С.

Предположим также, что необходимо вывести на экран ту же информацию, что и в предыдущем запросе: станция, год, месяц, средняя температура.

Чтобы не делать лишнюю работу, создадим новый запрос на основе старого.

Для этого в окне навигационной панели щелкнем правой кнопкой мыши на запросе Сортавала и в появившемся контекстном меню выберем строку Копировать. Затем щелкнем правой кнопкой в любом свободном месте окна и выберем в контекстном меню команду Вставить. В текстовое поле появившегося диалогового окна вставки введем имя нового запроса — пусть это будет Карелия, так как обе нужные станции находятся на территории Карелии.

Откроем новый запрос в режиме конструктора. Для этого достаточно щелкнуть правой кнопкой мыши на имени запроса и в контекстном меню выбрать строку Конструктор. В поле station ниже «Сортавала» введем название второй станции — Петрозаводск. Обратите внимание на заголовки строк: первая строка имеет заголовок Условие отбора, вторая — Или. Это «или» в данном случае относится ко всем строкам, расположенным ниже, то есть количество условий «или» не ограничено.

Поясним смысл данного условия. При отборе записей в запросе СУБД «просматривает» содержимое поля, по которому задан критерий отбора, и отбирает те записи, в которых значение данного поля *соответствует любому* из условий «или».

Теперь надо дополнить критерии отбора записей, указав интервал температур — от 10 до 15. В этом случае необходимо использовать оператор And (И), который указывает на диапазон выбора. Строка, соответствующая нашему примеру, выглядит так:

```
>10 And < 15
```

ВНИМАНИЕ Так как каждой станции соответствует отдельная строка, критерий отбора температур нужно повторить для каждой станции отдельно. В противном случае СУБД выберет записи по станции «Сортавала», отвечающие необходимому интервалу температур, и все записи по станции «Петрозаводск».

В некоторых случаях удобнее записывать условие «или» в одну строку, СУБД MS Access позволяет это делать. В нашем примере условие по станциям можно записать так:

```
"Сортавала" Or "Петрозаводск"
```

Естественно, в этом случае вторую строку с критерием по диапазону температур необходимо очистить, иначе Access выведет записи по *всем* станциям, отвечающие данному критерию.

Отметим, что наряду с условиями > и < (больше и меньше) в запросах можно также использовать условия >= (больше или равно), <= (меньше или равно), = (равно), <> (не равно). В среде MS Access 2002 последние два условия можно применять и к текстовым переменным в значении «соответствует» или «не соответствует».

Обратите внимание на то, что в нашем учебном примере обе станции относятся к региону Карелия, и других станций в данном регионе нет. Значит, для того чтобы отобрать эти две станции, можно добавить в запрос таблицу регионов и задать условие отбора Карелия.

Сделаем копию запроса Карелия1, назовем ее Карелия2 и откроем в режиме конструктора. Добавим таблицу Regions (для этого щелкните правой кнопкой мыши в верхней половине окна конструктора и в контекстном меню выберите команду Добавить таблицу) и заменим поле station таблицы Stations полем region таблицы Regions.

ВНИМАНИЕ Несмотря на то что ни одно поле таблицы Stations не выводится при выполнении запроса, сама таблица является его необходимым элементом, обеспечивая ссылочную целостность данных. Напомним, что все таблицы, участвующие в запросе, должны быть обязательно связанными.

Так как заранее известно, что в запросе отбираются записи по региону Карелия, вывод на экран названия региона представляется излишним. Сбросим флажок в строке Вывод на экран поля region. Окончательно запрос Карелия2 в окне конструктора выглядит так, как показано на рис. 2.8.

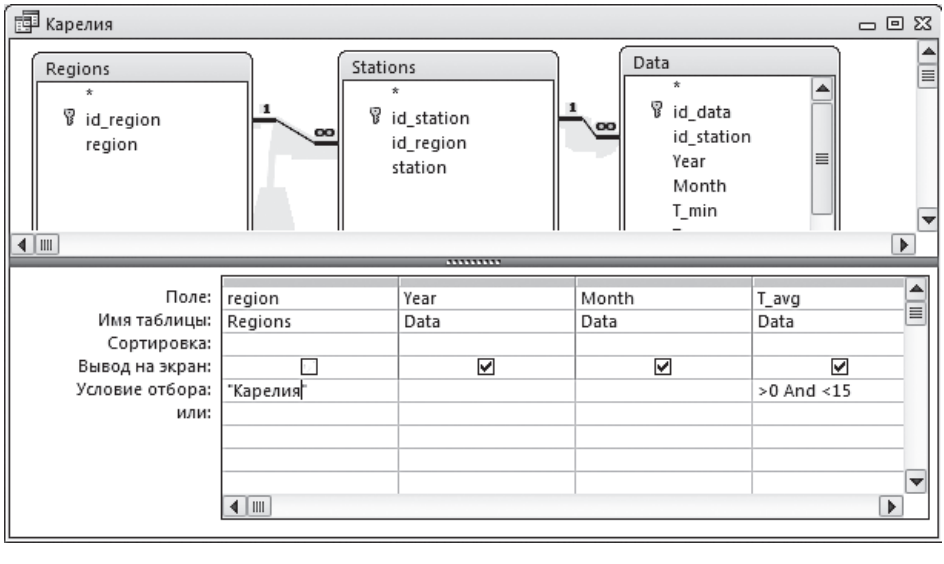

Рис. 2.8. Запрос Карелия2

2.1.3. Сортировка в таблице и запросе

Одним из способов, облегчающих нахождение нужной информации, является сортировка данных. Понятно, что если данные в таблице как-то упорядочены (по алфавиту, возрастанию или убыванию числовых значений, дате и т. д.), найти нужную запись гораздо легче.

В практической работе с БД приходится сортировать записи по разным столбцам (полям). В таблицах для этого используют инструмент Сортировка данных.

По умолчанию данные сортируются по значению ключевого поля, а в случае его отсутствия — в порядке ввода записей. Если требуется отсортировать записи по значению другого столбца, установите курсор в любое место внутри этого столбца и щелкните на одной из кнопок области фильтрации и сортировки: Сортировка по возрастанию или Сортировка по убыванию.

Для числовых полей и дат результат очевиден, текстовые поля в первом случае сортируются по алфавиту. Если текстовые поля содержат числовые символы, они сортируются именно как строки символов, то есть, например, 12 будет размещено впереди 2. Изменить ситуацию можно, добавив впереди 0 — если возникает необходимость сортировать такие поля в числовом порядке, число символов во всех записях должно быть одинаково.

Можно также щелкнуть в любой строке нужного столбца правой кнопкой мыши и выбрать соответствующие команды из контекстного меню.

Если в столбце, по которому производится сортировка, имеются пустые значения, при сортировке по убыванию они располагаются внизу таблицы.

При сохранении таблицы сохраняется и ее порядок сортировки.

Отметим, что нельзя сортировать данные по значениям полей гиперссылок, объектов OLE и полей MEMO.

Если возникает необходимость отсортировать данные по нескольким полям одновременно, нужно выделить эти столбцы и щелкнуть на кнопке Сортировка по возрастанию (убыванию). При этом столбцы должны располагаться рядом, и первыми сортируются те столбцы, которые расположены левее.

СОВЕТ Если предполагается сортировка данных в таблице по нескольким столбцам, лучше сразу располагать их в таблице в нужном порядке. Первоначальный порядок столбцов в таблицах лучше не менять. Следует помнить о том, что сортировать данные можно и в запросах, тогда порядок столбцов в таблицах не имеет значения.

Иногда при выполнении запроса требуется выполнить сортировку данных в некотором порядке, например вывести список сотрудников по алфавиту. В этом случае необходимо установить вид сортировки в специальной одноименной строке конструктора запросов. При этом соблюдается то же правило: при сортировке по нескольким полям одновременно более высокий приоритет имеет поле, расположенное левее.

Например, установим сортировку в запросе Сортавала по полям Year и Month. Так как поле Year расположено левее поля Month, записи выводятся отсортированными вначале по году, затем внутри года по месяцу, то есть поле, находящееся левее, имеет более высокий приоритет при выполнении данной операции. Количество полей сортировки не ограничено.

2.2. Вычисления в запросах

В данном разделе рассматриваются вычисления и использование выражений в запросах на выборку:

- основы использования арифметических, логических и текстовых выражений в запросах;
- использование встроенных функций MS Access;
- создание вычисляемых полей;
- использование параметров в запросе;
- использование статистических функций в групповых операциях;
- создание перекрестных запросов.

2.2.1. Использование выражений

Любой критерий отбора записей, например условие на значение поля таблицы, описанное ранее, является *выражением*. В частности, выражения используются при создании *вычисляемых полей*, то есть дополнительных столбцов запроса, в которые выводятся результаты вычисления по какой-либо формуле, а не данные из столбца таблицы. Такие вычисляемые поля могут создаваться также в формах или отчетах. Для создания сложных выражений нужно знать некоторые правила. Так, выражение может содержать константы, идентификаторы, функции и операторы.

- Константы содержат неизменные значения, которые, как правило, задаются при вводе данных. Например, любой код (идентификатор записи, например id_station) является константой. Константы широко используются для сравнения значений при выборке, например для выбора нужной метеостанции в рассмотренных ранее примерах. Существуют также именованные константы True, False, Yes, No, Null.
- Идентификаторы — это имена объектов БД, например имя таблицы или имена полей таблиц, которые при вычислении выражений возвращают текущие значения в этих полях, например значения температур воздуха.

ПРИМЕЧАНИЕ Если имя объекта содержит пробелы, идентификатор в выражении должен быть заключен в квадратные скобки. Таким образом, ввод выражений упрощается в случае, когда пробелы в названиях не используются.

- Функции возвращают в выражение некоторое значение. Аргументы функции заключаются в круглые скобки, которые необходимы даже в том случае, когда

аргументов нет. Известный в MS Office пример — функция `Date()`, возвращающая текущую дату.

- Операторы, или знаки арифметических и логических операций, а также операций присваивания, сравнения и конкатенации.

Использование констант уже не требует особых пояснений. Отметим только, что в Microsoft Access различаются два типа пустых значений: значения Null и пустая строка. Значение Null можно ввести в поле или использовать в выражениях и запросах для указания отсутствующих или неизвестных данных. Поля первичного ключа не могут содержать значения Null. Пустая строка — это строковое значение, не содержащее символов. Она используется в том случае, когда значение поля отсутствует. Для ввода пустой строки с клавиатуры следует ввести два символа прямых кавычек без пробела: `""`. В некоторых случаях пустое значение указывает на то, что могут существовать данные, которые пока неизвестны.

Например, если таблица с данными по экстремальным природным явлениям содержит пустое поле региона, это означает, что регион, где произошло это явление, пока неизвестен. В этом случае пустое поле имеет значение Null, указывающее, что точное значение неизвестно. Если значение атрибута в принципе отсутствует, можно ввести в поле пустую строку, указывающую на это обстоятельство.

Значение «пустая строка» определено только для полей таблиц с типом данных Текстовый, Поле MEMO или Гиперссылка.

Рассмотрим далее использование идентификаторов, функций и операторов. Любой объект БД имеет уникальное имя, по которому его всегда можно идентифицировать. В зависимости от положения в иерархии объектов это имя может содержать различное количество составных частей. В общем случае вверху иерархии находится объект Приложение, далее Таблица и Поле. Во всех приложениях MS Office ссылку на само приложение (MS Access) опускают. Так, поле станций таблицы Data в выражении может быть записано как `[Data]![Station]`.

Имя объекта можно записать непосредственно в соответствующую ячейку таблицы конструктора запросов. Иногда выражение с идентификатором удобнее строить с помощью Построителя выражений. Для того чтобы создать вычисляемое поле, необходимо щелкнуть правой кнопкой мыши в верхней строке пустого столбца и в контекстном меню выбрать строку Построить (или выбрать Построитель на вкладке Конструктор ленты). Так, на рис. 2.9 представлен пример обращения к таблице с помощью Построителя выражений. Двойной щелчок на поле station в средней области окна Построителя выражений вызывает появление имени объекта в верхней области.

Встроенные *функции MS Access* позволяют выполнять вычисления в запросах, формах и отчетах. Всего их около 300, и для удобства они сгруппированы по категориям: для работы с массивами, для преобразования типа данных, для работы с объектами базы данных и т. д. Так, на рис. 2.10 представлен пример выбора функции из категории Проверка, с помощью которой определяют, является ли значение поля числовым.

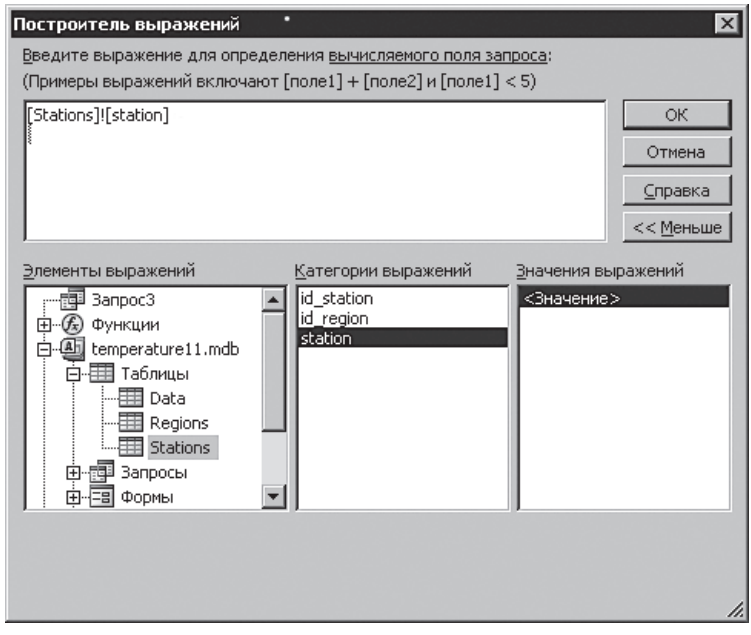

Рис. 2.9. Обращение к таблице из Построителя выражений

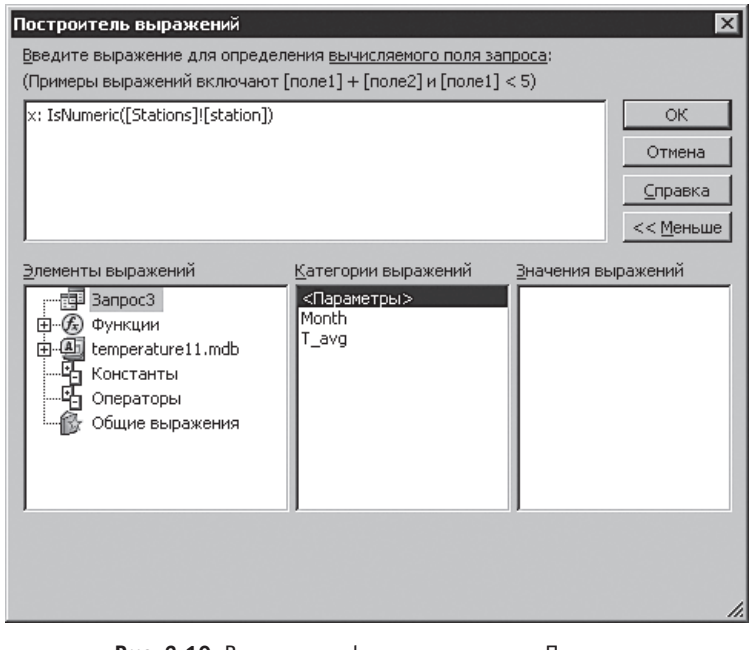

Рис. 2.10. Встроенные функции категории Проверка

При выполнении запроса в динамическую таблицу будет добавлено поле *x*, в которое будет выводиться 0, если значение не является числом, и 1, если это число.

Таким образом, чтобы создать выражение в запросе, следует записать в свободной верхней строке имя выражения, двоеточие и формулу.

При выделении функции в правом окне Построителя выражений можно получить справку по ней, щелкнув на круглой синей кнопке со знаком вопроса, расположенной вверху справа. Отметим, что если функция или объект не выводятся в нижней части Построителя выражений, это значит, что их использование недопустимо в той ситуации, в которой был вызван Построитель.

Приведем еще несколько примеров функций из различных категорий. Примером функции преобразования типов данных является функция `Val(stringexpr)`, которая переводит текстовую строку в число. Обратное преобразование выполняет функция `Str(number)`.

Функции работы с объектами баз данных используют для обращения к таблицам, запросам, формам и отчетам.

Функции даты и времени позволяют выполнять операции со значениями дат и времени, например вычислить промежуток в днях между двумя датами (`Datediff`).

Статистические функции возвращают статистические данные. Отметим, что многие статистические вычисления удобно делать с помощью групповых операций в запросах, которые рассматриваются в разделе 2.2.3. Данный вид функций подробнее обсуждается в разделе 2.4.2.

Примером функции обработки ошибок является `Error(errorcode)`, которая выводит сообщение об ошибке по ее номеру.

Функции сообщений ввода-вывода позволяют выводить осмысленные сообщения. Так, функция `=MsgBox("Подтвердите ввод";1;"Подтверждение")` выводит диалоговое окно, представленное на рис. 2.11.

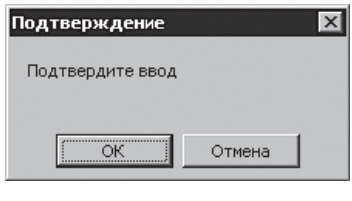

Рис. 2.11. Диалоговое окно функции MsgBox

В данном случае единица означает, что в диалоговое окно выводятся две кнопки ОК и Отмена, возвращающие при нажатии 1 или 2 соответственно. Полный список аргументов доступен в справке окна Построителя выражений.

Функции проверки возвращают логическое выражение, являющееся ответом на вопрос о типе аргумента. Так, если функция `IsNumeric ("varexpr")` возвращает константу `False`, ее аргумент не является числом.

Математические функции предназначены для выполнения математических операций, например вычисления экспоненты или косинуса числа.

Текстовые функции позволяют выполнять операции над строковыми переменными. Например, функция LCase(stringexp) переводит текстовую строку в нижний регистр.

Финансовые функции используются для вычисления различных финансовых параметров. Например, функция DDB(cost, salvage, life, period[, factor]) вычисляет снижение стоимости имущества на основе таких параметров, как начальная стоимость, стоимость реализации имущества, срок полезного использования, срок расчета и метод определения ставки.

Общие функции используются в программировании на VBA. Например, функция CodeDb() позволяет определить имя базы данных, в которой выполняется данный код.

Функции управления используют для выбора какого-либо варианта из нескольких. Так, функция IIf(Expr, Truepart, Falsepart) возвращает значение Truepart, если Expr = True, и значение Falsepart, если Expr = False.

Функции DDE/OLE используются для организации взаимодействия с другими приложениями.

Как уже отмечалось, полный список функций содержит около 300 наименований, поэтому более полное их описание в рамках настоящего пособия не имеет смысла. Следует отметить, что каждая функция имеет свою область применения, которую, в частности, можно определить с помощью справочной системы. Например, очевидно, что функцию MsgBox, о которой мы только что говорили, бессмысленно использовать в качестве вычисляемого поля запроса. В то же время перед выполнением запроса может возникнуть необходимость вывести какое-либо сообщение, и в этом случае данная функция может быть полезной. Задать такой порядок действий можно, например, с помощью макросов, которые рассматриваются в главе 5. Наиболее полное использование функций возможно с помощью встроенного языка программирования Visual Basic.

Со списком функций и их кратким описанием можно ознакомиться в приложении 5. Можно также воспользоваться справочной системой MS Access.

Отметим, что некоторые востребованные вычислительные операции, например групповые операции, воспроизводятся с помощью визуального конструктора запросов (см. раздел 2.2.3).

Арифметические операторы выполняют основные арифметические действия, их список приведен в табл. 2.1.

В MS Access поддерживается шесть *логических операторов*:

- And — логическое И;
- Or — логическое ИЛИ;
- Not — логическое отрицание;
- Xor — исключающее ИЛИ;
- Eqv — логическая эквивалентность;
- Imp — логическая импликация.

Так, если в поле условий какого-либо столбца записать `Is Not Null`, в результирующее множество попадут только записи с непустыми атрибутами этого поля. Результаты использования логических операторов систематизированы в табл. 2.2.

Таблица 2.1. Арифметические операторы

Оператор	Пример	Описание
-	[T_max]-[T_min]	Разность
+	[T_med]+2	Сумма
- (унарный)	-5	Смена знака операнда
*	2*2	Произведение
/	[Precip]/10	Частное
\	[Precip]\10	Частное целочисленное
^	[]^[power]	Основание в степени power

Таблица 2.2. Результаты использования логических операторов

Выражение	X = True Y = False	X = True Y = True	X = False Y = False	X = False Y = True
X And Y	False	True	False	False
X Or Y	True	True	False	True
Not X	False	False	True	True
X Xor Y	True	False	False	True
X Eqv Y	False	True	True	False
X Imp Y	False	True	True	True

Если значение одного из операторов Null, то во всех случаях, кроме использования оператора `Imp`, значение результата также будет Null.

Оператором присваивания является знак равенства. Его можно использовать также как *оператор сравнения* в условиях отбора запроса. Для сравнения численных операндов можно использовать также символы <, >, >=, <= и <>. Для сравнения текстовых операндов чаще всего используют оператор `Like`. Так, запись в строке условий:

```
Like "Пет*"
```

приведет к выбору из таблицы метеостанций значений «Петербург» и «Петрозаводск». Как и в целом в ОС Windows, символ * означает любую последовательность символов, а символ ? — один произвольный символ. Такие символы называются *групповыми*. Кроме этого, используется знак подчеркивания «_», также замещающий любой одиночный символ, и знак процента, замещающий любое число символов, включая 0. Для объединения двух текстовых переменных используют *оператор конкатенации*, или *слияния строк*, обозначаемый &. Отметим, что иногда использование символа + приводит к тому же результату, тем не менее использование специального оператора является более корректным.