8. Особенности запросов

Вычисляемое поле – поле, определенное в запросе, но не существующее в таблице. Программа Access вычисляет значение этого поля, основываясь на одном или нескольких других полях таблицы. Значения вычисляемых полей никогда не хранятся в БД – программа генерирует их при каждом выполнении запроса.

Определение вычисляемого поля

Для создания вычисляемого поля следует задать два компонента: имя поля и выражение, определяющее вычисление, которые должна выполнить программа Access. Вычисляемые поля определяются с помощью следующего состоящего из двух частей шаблона:

ИмяВычисляемогоПоля: Выражение

Например, поле *Цены с налогом* определяется следующим образом: *ЦенаСНалогом: [Цена] *1.10*

По сути, это выражение сообщает программе Access о том, что нужно взять поле *Цена* и умножить его на 1.10, что эквивалентно повышению цены на 10%. Access повторяет это вычисление для каждой записи, входящей в результаты запроса. Для того чтобы это вычисление выполнялось, в таблице должно существовать поле *Цена*. Но вовсе необязательно отображать отдельно это поле в окне результатов запроса.

Можно сослаться на поле *Цена*, используя его полное имя, состоящее из имени таблицы с последующей точкой, за которой указано имя поля: *ЦенаСНалогом: [Товары].[Цена] * 1.10*

Для добавления вычисляемого поля *Цена с налогом* понадобится *Конструктор*. Сначала следует найти столбец, в который нужно вставить вычисляемое поле. Обычно оно добавляется и в конец, и в первый свободный столбец, хотя можно раздвинуть существующие столбцы и освободить для него место. Далее в ячейке *Поле* нужно ввести полное определение поля (рис. 8.1).

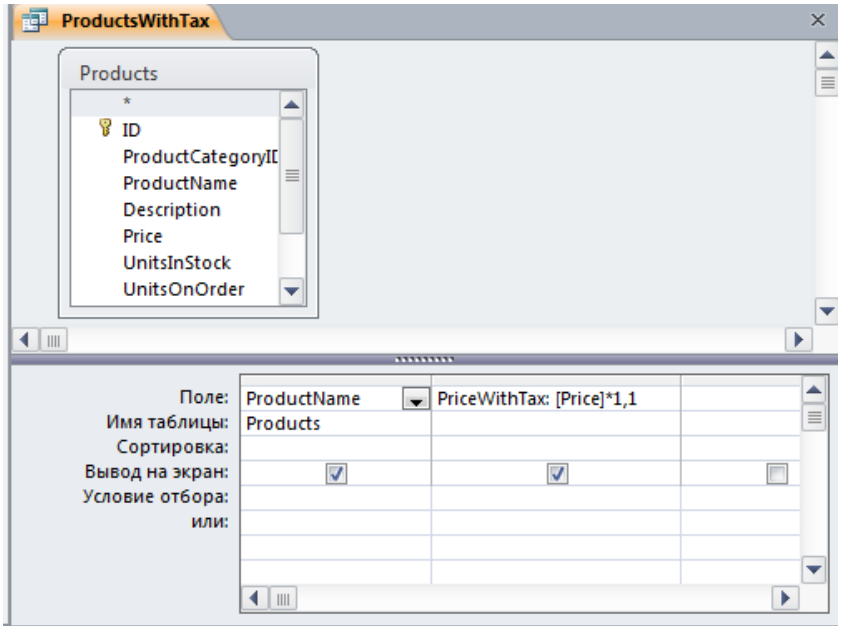

Рис. 8.1. Запрос отображает два поля из БД и вставляет вычисляемое поле PriceWithTax.

У вычисляемых полей есть одно ограничение – поскольку информация не сохраняется в таблице, нельзя их редактировать. Если нужно изменить цену, необходимо отредактировать базовое поле Цена (*Price),* так как нельзя корректировать поле *ЦенаСНалогом (PriceWithTax).*

Выражение в каждый конкретный момент времени обрабатывает отдельную запись. Если нужно объединить информацию из некоторых записей для вычисления итогов или средних значений, необходимо использовать свойства группировки.

Обычное поле *Цена*, которое программа Access применяет для вычисления поля *ЦенаСНалогом PriceWithTax*, вообще не отображается (рис.8.2).

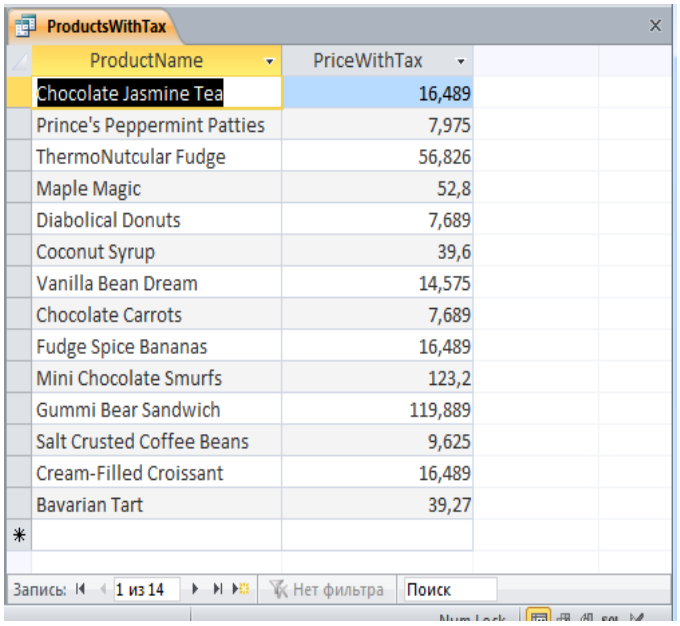

Рис. 8.2. Результаты запроса

Результаты отображают поле *PriceWithTax* с надбавкой 10%. Главное состоит в том, что вычисляемая информация теперь доступна постоянно, несмотря на то, что она не хранится в БД.

Простая математическая обработка числовых полей

Многие вычисляемые поля полагаются на обычные арифметические операции. В табл. 8.1 представлен краткий обзор основных вариантов комбинирования чисел.

Таблица 8.1.

Арифметические операции

Операция	**Название**	**Пример**	**Результат**
+	Сложение	1+1	**2**
-	Вычитание	**1-1**	**0**
*	Умножение	**2*2**	**4**
^	Возведение в степень	**2^3**	**8**
/	Деление	**5/2**	**2.5**
\	Деление нацело (возвращает наименьшее целое число и отбрасывает остаток)	**5\2**	**2**

Mod	Остаток от деления (возвращает остаток, полученный в результате деления нацело)	**5** Mod **2**	**1**

Для создания выражения можно использовать произвольное количество полей и операций.

Поля с датами

Операции сложения и вычитания можно применять к полям, содержащим даты. Можно применять и умножение, деление и все что угодно, но реального смысла такие значения иметь не будут.

Применяя операцию сложения, можно добавить обычное число к полю с датой. Это число сдвигает дату вперед на указанное число дней. Далее приведен пример, добавляющий дополнительные две недели к сроку платежа, установленному компанией: *ExtendedDeadline: [DueDate] + 14*

Если это выражение применить к дате 10 января 2012 г., новая дата будет 24 января 2012 г.

Используя операцию вычитания, можно найти число дней между двумя датами. Далее показано, как вычисляется период между временем помещения заказа и временем доставки: *ShippingLag: [ShipDate] - [OrderDate]*

Если доставка произошла через 12 дней после того, как сделан заказ, будет значение 12.

Поля типа Дата/время включают информацию о времени суток. В вычислениях данные о времени представляются как дробная часть числа. Если выполнено вычитание одной даты из другой и получено число 12.25, оно представляет период 12 дней и 6 часов (поскольку 6 часов равно 25% суток).

Если нужно включить фиксированные даты в запросы, их необходимо обрамлять символами # и использовать формат *Месяц/День/Год*. Далее приведен пример, использующий такой подход для вычисления количества дней между датой своевременного предоставления задания (20 марта 2012 г.) и датой действительного его предоставления студентом:

LateDays: [DateSubmitted] - #03/20/12#

Положительное значение указывает на то, что значение в поле *DateSubmitted* больше (более поздняя дата), чем предельный срок сдачи – студент опоздал. Отрицательное значение указывает на то, что студент сдал работу раньше назначенного в расписании срока.

Порядок выполнения операций

Если в выражении длинная строка вычислений, программа Access следует строгим правилам определения старшинства операций или, говоря математическим языком, учитывает, какое вычисление выполняется первым при наличии нескольких вычислений в выражении. Итак, если выражение длинное, Access не просто обрабатывает его слева направо, а оценивает выражение фрагмент за фрагментом в следующем порядке:

- скобки;
- проценты;
- возведение в степень;
- деление и умножение;
- сложение и вычитание.

Выражения с текстовыми значениями

Обычно вычисляемые поля имеют дело с числовыми данными, но так бывает не всегда, существуют действительно удобные способы обработки текста.

Можно слить текст. Есть возможность связать несколько полей с адресной информацией и отображать их все в одном поле, экономя пространство и, возможно, облегчая экспорт этих данных и другую программу.

Для слияния текста применяется оператор амперсанд (*&*). Далее показано, как создать поле *FullName (ФИО)*, в котором собрана информация из полей *FirstName и LastName*:

FullName: [FirstName] & [LastName]

Это выражение выглядит достаточно корректным, но на самом деле у него есть один недостаток. Поскольку не вставлено никаких пробелов, имя и фамилия в результате прижаты друг к другу, например, так: *BenJenks*. Лучше слить три фрагмента текста: имя, пробел и фамилию. Вот исправленная версия: *FullName: [FirstName] & " " & [LastName]*

Приведенное выражение создаст значение: Ben Jenks. Можно также поменять местами имя и фамилию и отделить их запятой, например, *Jenks, Be*n для облегчения сортировки:

FullName:[LastName]&","&[FirstName]

В программе Access есть два типа текстовых значений: те, которые извлекаются из других полей, и те, которые вводятся непосредственно (или фиксированные). Когда вводятся текстовые константы, такие как запятая или пробел в предыдущем примере, их следует заключать в кавычки, чтобы программа Access знала, где начинается и заканчивается текст.

Можно применять амперсанд (&) для сцепления текста с числовыми значениями. Если нужно, чтобы слегка бесполезный текст "*Цена:*" появлялся перед каждым значением цены, нужно применить следующее вычисляемое поле:

Цена: "Цена:" & [Цена]

Построитель выражений

Для быстрого поиска нужных функций Access предлагает компонент, именуемый *Построителем выражений*. Для его запуска выполнить следующие действия:

- Открыть запрос в *Конструкторе*.
- Щелкнуть правой кнопкой мыши поле, в которое нужно вставить выражение, и выбрать команду *Построить*.

Если создается вычисляемое поле, нужно щелкнуть правой кнопкой мыши в ячейке *Поле.* Если создается условие отбора, следует щелкнуть правой кнопкой мыши в ячейке *Условие отбора*. После выбора команды *Построить* на экране появляется окно *Построителя выражений*, отображающее текущее содержимое поля (рис. 8.3).

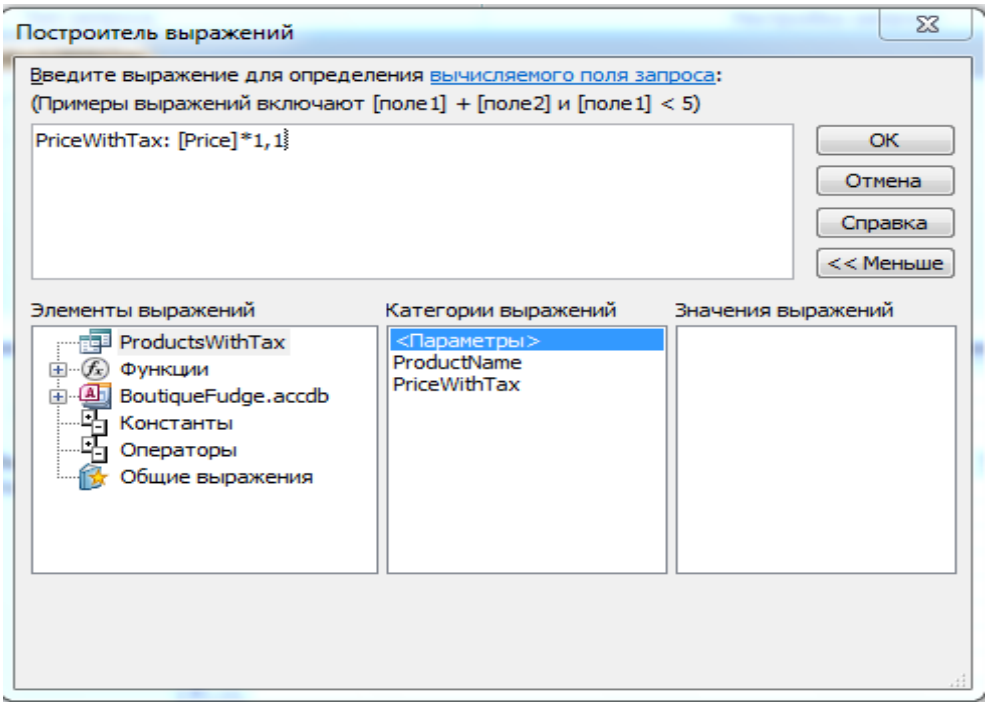

Рис. 8.3. Построитель выражений

Построитель выражений состоит из текстового поля в верхней части окна, в котором можно редактировать выражение, кнопок быстрой вставки знаков обычных операций таких как +, -, / и *, если почему-либо их не найти на клавиатуре, и трехпанельного обозревателя в нижней части окна, который поможет найти нужные поля и функции.

- Вставить или отредактировать выражение. У *Построителя выражений* есть два ускоряющих приема работы. Можно вставлять имя без ввода с клавиатуры (рис.8. 4) и можно найти функцию с помощью обзора (рис. 8.5). *Построитель выражений* – универсальное средство создания выражений в вычисляемых полях и условиях отбора. Некоторые параметры имеют смысл только в одном из его назначений. Логические операторы, такие как символ равенства *(=), And, Or, Not и Like*, удобны для задания условий фильтрации, но бесполезны в вычисляемых полях.

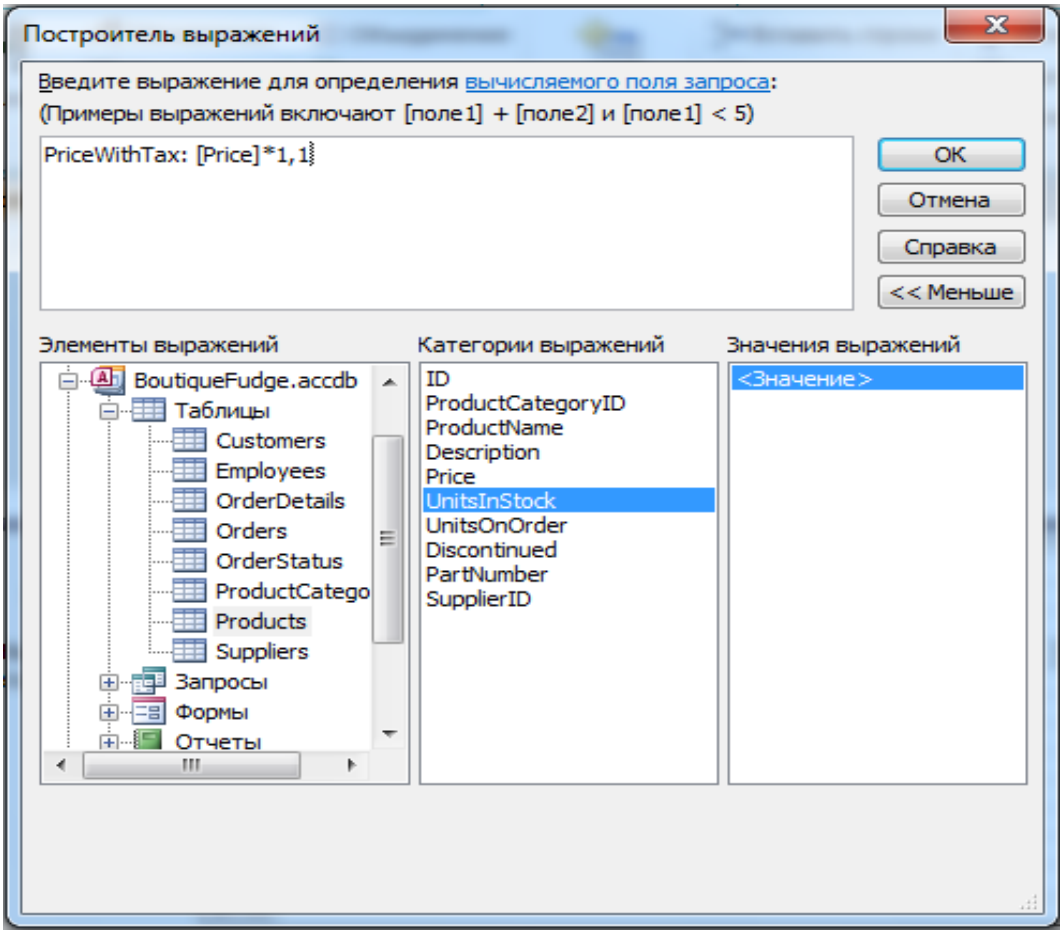

Рис. 8.4. Вставка дополнительного имени поля

Для вставки имени поля следует дважды щелкнуть кнопкой мыши папку *Таблицы* в самом левом списке. Затем щелкнуть мышью вложенную папку, соответствующую нужной таблице. Дважды щелкнуть кнопкой мыши имя поля в среднем списке для добавления его в выражение.

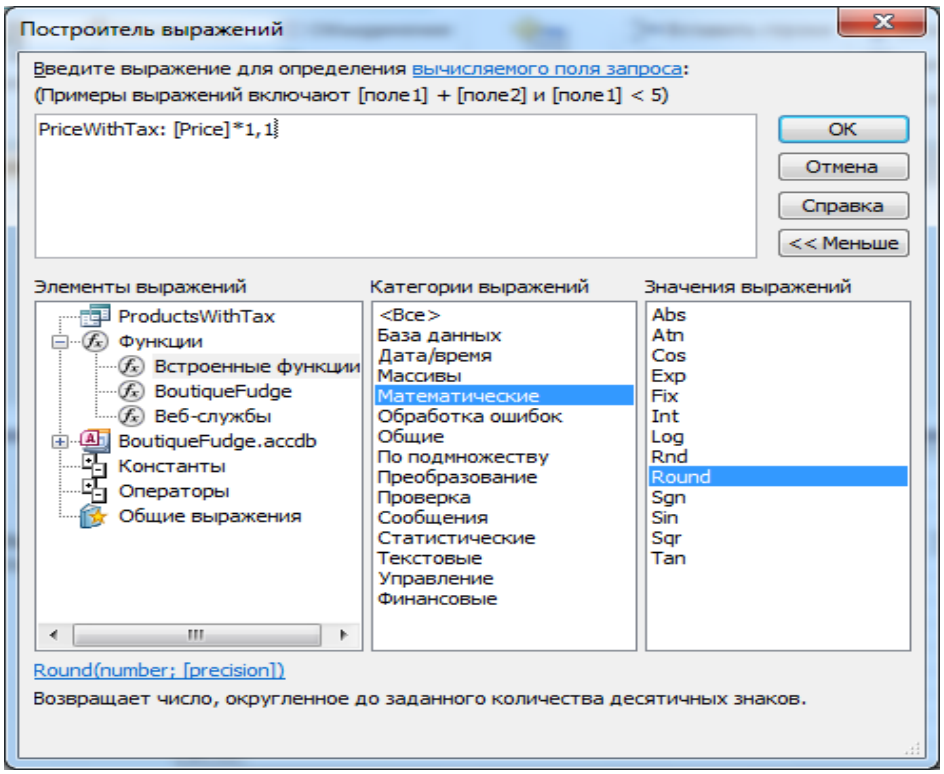

Рис. 8.5. Использование встроенных функций в построителе выражений

Поиск функции нужно начать с двойного щелчка кнопкой мыши папки *Функции* в левом списке. Затем выбрать вложенную папку *Встроенные функции*. Другие варианты отображают любые пользовательские функции, которые были добавлены в БД с помощью пользовательского кода *VBA*. Далее выбрать категорию функции в среднем списке. В правом списке показаны все функции в выбранной категории. Можно дважды щелкнуть кнопкой мыши функцию для вставки ее в выражение. При вставлении имен полей в *Построителе выражений*, они записываются в более длинном формате с обязательным указанием имени таблицы. Можно увидеть *[Products] ! [Price]* вместо просто [*Price*].

- Щелкнуть мышью кнопку *ОК*. Программа Access скопирует новое выражение в ячейку *Поле* или *Условие отбора*. Когда в *Построителе выражений* вставляется функция, программа добавляет заполнители (например, <*number*> и <*precision*>), на место которых нужно ввести аргументы. Следует заменить этот текст нужными значениями. В справке функции округления Round () описываются ее назначение и два параметра. Второй параметр – количество знаков в дробной части – заключен

в квадратные скобки, что означает необязательное значение. Если пропустить его, то программа Access округлит значение до ближайшего целого числа. Слева приведен перечень функций, который позволяет просмотреть любую другую функцию Access и прочесть ее описание.

Форматирование чисел

Format() – интересная математическая функция, преобразующая число в текст. Она интересна, потому что создаваемый текст можно отформатировать несколькими способами, управляя, таким образом, представлением чисел.

В табл. 8.2. приведены возможные варианты форматирования.

Таблица 8.2.

Варианты форматирования

Формат	**Описание**	**Пример**
Денежный	Выводит число с двумя знаками в дробной части, разделителями для тысяч и знаком валюты	$1 433.20
Фиксированный	Отображает число с двумя десятичными знаками	1433.20
Основной	Выводит на экран число с двумя десятичными знаками и разделителями тысяч	1 433.20
Процентный	Отображает процентное значение (число, умноженное на 100, и со знаком процента). Выводит две цифры справа от десятичной точки	143320.00%
Экспоненциальный	Отображает число в научной нотации с двумя десятичными знаками	1.43E+03
Да/нет	Отображает Нет, если число равно 0, и Да, если число отлично от 0. Можно использовать аналогичные типы формата Истина/Ложь и Вкл/Выкл	Да

Хитрость применения функции Format() состоит в выборе текста, задаваемого в качестве второго аргумента для получения желаемого результата.

Для того чтобы понять разницу, вернемся к выражению, использованному ранее для снижения цены: SalePrice:[Price]*0.95

Даже если у поля *Price* не *Денежный* тип, вычисленные значения в поле *SalePrice* (продажная цена) выводятся как обычные числа (без знака валюты, разделителя тысяч и т. д.). Решить эту проблему можно, применив функцию Format() для задания денежного формата вывода:

SalePrice: Format([Price] / 0.95, "Currency")

Теперь вычисленные значения содержат знак валюты. Более того, поскольку денежные суммы отображаются с двумя знаками после точки, не нужно больше применять функцию *Round ( )*.

Дополнительные математические функции

Математическим функциям в программе Access не уделяется должного внимания, потому что потребность в них возникает крайне редко. Функции *Round()* и *Format()* – самые полезные в этой категории – но есть еще несколько других (табл. 8.3), к которым знатоки Access обращаются иногда в вычисляемых полях.

Таблица 8.3.

Функции для числовых данных

Функция	**Описание**	**Пример**	**Результат**
Sqr(x)	Извлекает квадратный корень	Sqr(9)	3
Abs(x)	Возвращает положительное значение (отрицательные числа становятся положитель-ными)	Abs(-6)	6
Round()	Округляет число до заданного числа десятичных знаков	Round (8.89, 1)	3.9

Fix(x)	Возвращает целую часть числа, отбрасывая любую дробную часть	Fix(8.89)	8
Int(x)	То же что функция **Fix**(), но отрицательные числа округляются до ближайшего меньшего целого числа, а не большего	Int(-8.89)	-9
Rnd()	Генерирует случайное дробное число в диапазоне	Int((6)* Rnd+1)	Случайное
Format()	Преобразует число в форматированную текстовую строку в соответствии с выбранными параметрами	Format(24,6 ,Currency)	$243.60
Val (x)	Преобразует числовые данные в текстовом поле в настоящее число так, что вы можете использовать его в вычислении. Останавливается, как только находит нецифровой символ, и возвращает 0, если не найдено ни одной цифры	Val ("315 Crossland St")	315

Текстовые функции

С текстом тоже можно делать многое. В целом есть три способа обработки текста:

- Слияние текста. Можно соединить несколько текстовых полей в одном. Для этого способа не нужна функция – достаточно оператора &.
- Извлечение подстроки из текстовой строки.
- Замена строчных букв прописными и наоборот.

В табл. 8.4 перечислены функции, наиболее часто применяемые для обработки текста.

Таблица 8.4.

Функции для работы с текстом

Функция	**Описание**	**Пример**	**Результат**
UCase()	Выводит текст прописными буквами	UCase("Hi There")	HI THERE
LCase()	Выводит тест строчными буквами	LCase("Hi There")	hi there
Left()	Выводит заданное число символов, начиная от левого края строки	Left("Hi There", 2)	Hi
Right()	Выводит заданное число символов, начиная от правого края строки	Right ("Hi There", 5)	There
Mid()	Выводит часть строки, начиная с заданной позиции, и заданное число символов	Mid ("Hi There", 4, 2)	Th
Trim()	Удаляет пробелы с обеих сторон (или используйте LTrim ( ) и RTrim( ) для удаления пробелов только в начале или конце строки)	Trim(" Hi There ")	Hi There
Len()	Подсчитывает количество символов в текстовой строке	Len("Hi There")	8

С помощью этих функций можно создать вычисляемое поле, которое отображает фрагмент длинной текстовой строки или изменяет вид отображения (строчные или прописные буквы).

Применение текстовых функций в условиях отбора не столь очевидно. Можно создать условие фильтрации, задающее совпадение с частью текстовой строки, а не со всей строкой. Далее приведен пример условия отбора, выбирающего записи, начинающиеся с *"Choco": Left([ProductName], 5) = "Choco"*

Функция Len(S) – особый случай. Она проверяет текстовое значение и возвращает числовую информацию (в данном случае количество символов в строке, включая все пробелы, буквы, цифры и специальные символы). Эта функция не слишком полезна в простых вычисляемых выражениях, т. к. редко будет интересовать количество букв в текстовой строке. Но она позволяет создавать интересные условия отбора, включая, например, такое, которое отбирает все записи с полем *Description* короче 15 символов: *Len(Description) <15.*

Функции для обработки дат

Несомненно, многие применяют функции *Now()* и *Date()*. Эти функции извлекают текущие дату и время или только текущую дату. Их можно применять в запросах, работающих с заказами, принесшими доход в текущем году.

Вот условие для выбора просроченных проектов: =<Date()

Вставить его в ячейку *Условие отбора* поля *DueDate(срок платежа)* и будут видны только те записи, в которых поле *DueDate* содержит дату, наступившую ранее нынешнего дня.

Анализ дат может быть более сложным в сочетании с функцией *DatePart()*, которая извлекает часть информации из даты. *DatePart()* может определить номер месяца или год, позволяя игнорировать другие подробности (такие как число или время). С помощью *DatePart()* и *Date()* можно легко написать условие фильтрации, отбирающее заказы, сделанные в текущем месяце: *DatePart("m", [DatePlaced])=DatePart("m", Date()) And* DatePart("yyyy", [DatePlaced])=DatePart("yyyy", Date())

Это довольно длинное выражение на самом деле представляет собой комбинацию двух условий, соединенных ключевым словом *And*. Первое условие сравнивает номер месяца текущей даты с датой, хранящейся в поле *DatePlaced*:

DatePart("m", [DatePlaced])=DatePart ("m", Date( ))

Приведенное выражение устанавливает, что у обеих дат один и тот же календарный месяц, но необходимо также убедиться в том, что год у них тоже совпадает:

DatePart("yyyy", [DatePlaced])=DatePart("yyyy", Date ( ))

Сложность применения функции *DatePart()* (и некоторых других функций для дат) заключается в понимании идеи компонентов, составляющих дату. Применяя символ m в функции *DatePart()*, выдается номер месяца, а используя текст yyyy, извлекается четерехсимвольный номер года. В табл. 8.5 приведены все возможные варианты.

Таблица 8.5.

Компоненты даты

Компонент	**Описание**	**Значение на 20 февраля, 2012 г. 1:30 PM**
yyyy	Год в четырехсимвольном формате	2012
q	Квартал от 1 до 4	1
т	Месяц от 1 до 12	2
y	День в году, от 1 до 365 (обычно)	51
d	День в месяце от 1 до 31	20
w	День недели, от 1 до 7	2
ww	Неделя в году, от 1 до 52	8
h	Час, от 1 до 24	13
n	Минута, от 1 до 60	30
s	Секунда, от 1 до 60	0

Вычисления для дат и времени

Используя функции для дат, всегда следует помнить о датах, содержащих информацию о времени (все даты могут содержать данные о времени суток). Но, выбирая подходящий формат для поля с датами, программе Access сообщается о том, нужно ли отображать временной компонент даты и разрешать пользователям

вводить его. Чаще всего можно пользоваться форматом, скрывающим любую информацию о времени суток.

Проблема: функция Date() возвращает текущую дату со значением времени суток, равным 0. Другими словами, если текущая дата – 4 июля 2012 г., то функция Date() возвращает первую секунду 4 июля 2012 г. – момент, когда часы показывают полночь (12:00 a.m.)

Если не хранить значения времени суток, эта проблема не важна, поскольку у всех дат время суток равно 0. Но что произойдет, если использовать *Полный формат даты* в поле *DueDate*, разрешающий пользователям вводить и дату, и время. Теперь у условия отбора *=<Date()* несколько иной смысл –Access сообщается о необходимости отобрать, как совпадающие, все поля с датами, наступившими до первой секунды текущего дня. Это условие не выберет запись со сроком платежа, назначенным на 16:00 текущего дня.

В данной ситуации, возможно, нужно изменить условие отбора на следующее: *<(Date()+1)*. *Date()+1* – это завтра. Другими словами, это условие отбирает любые записи со сроком платежа, истекшим до первой секунды завтрашнего дня.

Между прочим, у программы Access есть функция *Now()*, возвращающая текущую дату и время. Таким образом, следующее условие фильтрации отбирает все записи, соответствующие текущему моменту (текущего дня) или любому моменту времени во все предшествующие дни: *=<Now()*.

Компоненты дат применяются в нескольких функциях обработки дат, включая функции *DatePart()*, *DateAdd()* и *DateDiff()*. В табл. 8.6 приведены эти и дополнительные полезные функции, относящиеся к датам.

Таблица 8.6.

Функции обработки дат

Функция	**Описание**	**Пример**	**Результат**
Date ()	Получает текущую дату	Date ()	1/20/2012
Now ()	Получает текущую дату и время	Now ( )	1/20/2012 10:16:26 PM

DatePart()	Извлекает часть даты (например, год, месяц или день в месяце)	DatePart(#1/20/ 2012#,"d")	20
DateSerial()	Преобразует год, месяц и день в значение даты Access	DateSerial(2012,5,4)	5/4/2012
DateAdd()	Сдвигает дату на заданный интервал	DateAdd ("yyyy", 2, #22/ll/2012#)	22/11/2012
DateDiff()	Определяет интервал между двумя датами	DateDiff("w", #10/l5/2011#, #l/ll/2012#)	12
MonthName()	Получает название, соответствующее номеру месяца (от 1 до 12)	MonthName (1)	"January"
WeekdayName()	Получает название, соответствующее номеру дня в неделе (от 1 до 7)	WeekdayName (1)	"Sunday"
Format()	Преобразует дату в форматированный текст (используя любой формат даты, описанный в табл. 2.3)	Format(#27/04/ 2012#, "Long Date")	"April 27, 2012"

У программы Access есть другие функции обработки дат, выполняющие часть алгоритма функции *DatePart()*. Примером может служить функция *Month()*, извлекающая номер месяца из даты. К другим аналогичным функциям относятся *Year()*, *Day()*, *Hour()*, *Minute()* и *Second()*. Эти функции не дают никаких преимуществ, но можно их встретить в запросах других пользователей, применяющих их для получения аналогичных результатов.

Обработка пропущенных или неопределенных значений

В БД есть два типа полей: обязательные и необязательные. Как правило, в БД поля необязательные, что означает для неаккуратного пользователя возможность пропуска большого числа значений. Эти пропущенные значения называют неопределенными (*null*), и их следует тщательно обрабатывать.

Если нужно написать условие отбора, вылавливающее неопределенные значения, просто ввести в ячейку *Условие отбора* следующий текст: *Is Null*

Это условие выберет все записи с пропущенными значениями. Если игнорировать несвязанные записи, то нужно заменить условие отбора на обратное: *Is Not Null*

Иногда не нужно специально искать (или игнорировать) неопределенные значения. Вместо этого надо заменить их для рассматриваемой задачи чем-то более информативным. К счастью, как раз для этого есть функция со странным названием *Nz()*. У функции *Nz()* два аргумента. Первое значение, как правило, поле запроса может содержать неопределенное значение. Второй параметр – это значение, которое нужно отобразить в результатах запроса, если программа Access найдет неопределенное значение. Пример, использующий функцию Nz() для преобразования в 0 неопределенных значений в поле *Quantity: Nz([Quantity],0)*

Преобразование в 0 – стандартное поведение для функции *Nz()*, поэтому можно опустить второй параметр, если это как раз то, что нужно: *Nz([Quantity])*

Эта функция жизненно важна, если нужно создать вычисляемые поля, обрабатывающие значения, которые могут быть неопределенными. Рассмотрим кажущийся безобидным пример: *OrderItemCost:[Quantity]*[Price]*

Это выражение приведет к ошибке, если значение поля *Quantity* будет неопределенным. Если у одного из операндов в вычислении значение неопределенное, результат автоматически становится неопределенным. В данном примере это означает, что поле *OrderItemCost* для данной записи становится неопределенным. Хуже того, если *OrderItemCost* ввести в другое вычисление или промежуточный итог, они тоже станут неопределенными. Прежде чем об этом станет известно, все значимые данные запроса превратятся в кучу ячеек с неопределенными значениями.

Для устранения этой проблемы следует очистить необязательные ноли от неопределенных значений с помощью функции Nz(): *OrderItemCost:Nz([Quantity])*Nz([Price])*

Наконец, функцию *Nz()* можно использовать для замены всех неопределенных значений другой величиной. В текстовое поле

можно ввести что-то более информативное. Пример, отображающий текст ("*Not Entered*"– не введено) рядом с каждой записью, не содержащей имени и фамилии:

Name:Nz([FirstName]&[LastName],"[Not Entered]")

Итоговые данные

Все запросы, которые применялись до этого момента, имели дело с отдельными записями. Если выбирались *N* записей из таблицы, например, *Клиенты*, то можно было увидеть все *N* записей в результатах. Также можно группировать записи для получения промежуточных и общих итогов. В этом случае легче просматривать большие объемы информации и делать важные далеко идущие выводы.

Далее приведены примеры полезных подводящих итоги запросов:

- подсчет студентов в каждой группе;
- подсчет количества заказов, сделанных каждым клиентом;
- сумма, потраченная на один продукт;
- общая сумма долга или платежа клиента;
- подсчет среднего объема заказа, сделанного каждым клиентом;
- поиск самого дорогостоящего или самого дешевого заказа, сделанного клиентом.

Эти операции – подсчет, суммирование, определение среднего и поиск максимального и минимального значений – основные варианты в *итоговом запросе*. Итоговый запрос – это вид запроса, который должен перелопатить большое количество записей и выдать лаконичные итоги.

Для создания итогового запроса выполнить следующие действия:

- Создать новый запрос, выбрав *Создание* → *Другие* → *Конструктор запросов*.
- Добавить нужные таблицы с помощью диалогового окна *Добавление таблицы* и щелкнуть мышью кнопку *Закрыть*.
- Вставить поля, которые нужно использовать.
- Выбрать *Работа с запросами* → *Показать или скрыть* → *Итоги*.

Программа Access вставляет ячейку *Групповая операция* для каждого поля сразу под ячейкой *Таблица*. Для каждого поля задайте вариант из списка Групповая операция. Этот вариант определяет использование поля для вычисления итога или для группировки.

Итоговый запрос немного отличается от обычного запроса. Каждое поле должно попадать в одну из следующих категорий:

- Поле используется в итоговом вычислении таком как определение среднего, подсчет количества и т. д. Тип нужного вычисления выбирается с помощью ячейки *Групповая операция*. В табл. 8.7 перечислены все варианты из ячейки *Групповая операция*.

Таблица 8.7.

Варианты получения итоговых данных в ячейке *Групповая операция*

Значение	**Описание**
Группировка	Группирует записи, основываясь на значениях в этом поле
Sum	Суммирует значения этого поля
Avg	Находит среднее для значений этого поля
Min	Возвращает наименьшее значение в этом поле
Max	Возвращает наибольшее значение в этом поле
Count	Подсчитывает количество записей (независимо от того, какое поле вы используете)
First	Возвращает первое значение в этом поле
Last	Возвращает последнее значение в этом поле

В табл. 8.7 не указаны два варианта, предназначенные для статистиков – StDev и Var – которые вычисляют стандартное отклонение и дисперсию ряда чисел.

- Поле применяется для группировки. Обычно итоговые запросы соединяют в большой общий итог. Но можно разбить результаты на более мелкие промежуточные итоги.
- Поле используется для фильтрации или отбора. В этом случае в ячейке *Групповая операция* нужно выбрать

Условие. Нужно также сбросить флажок *Вывод на экран*, поскольку программа Access не может выводить отдельные значения в итоговых сводках.

Если сделать попытку вставить в итоговый запрос поле, которое не используется для вычисления или группировки и не скрыто, то произойдет ошибка при попытке выполнить запрос.

В примере (рис. 8,6) в поле *Price* применяются три разные групповые операции: *Max, Min* и *Avg*.

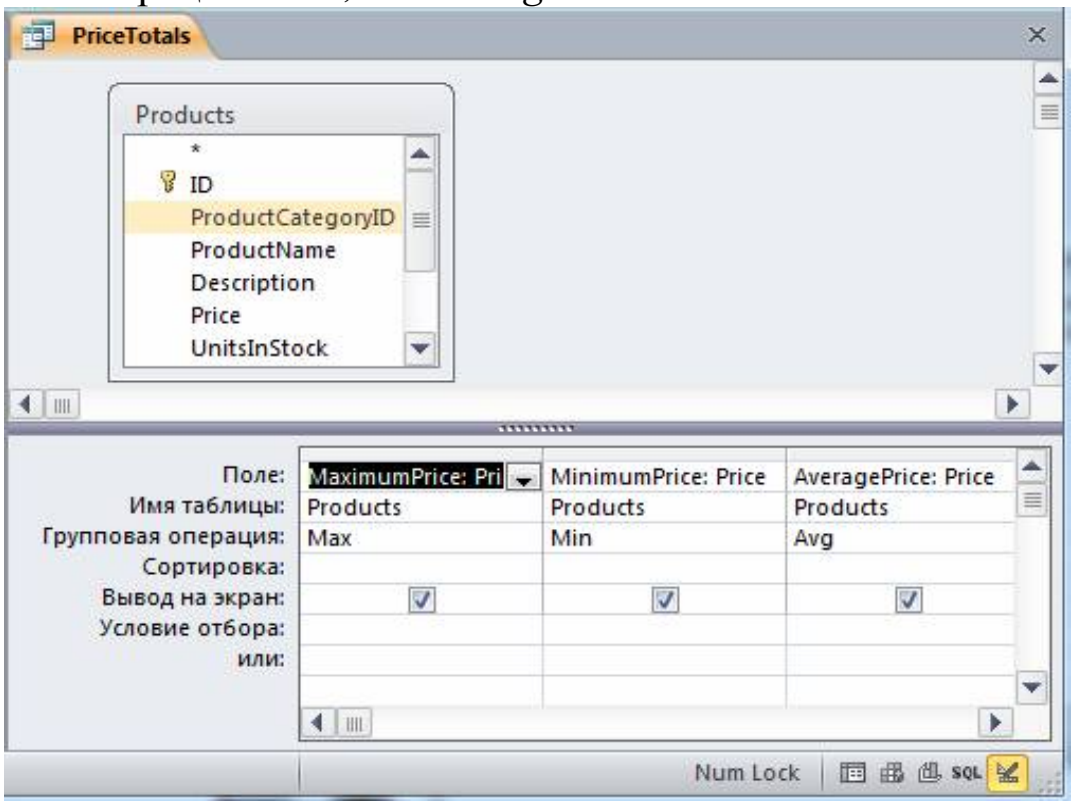

Рис.8.6. Создание итогового запроса (режим конструктора).

В данный итоговый запрос поле *Price* включено трижды и использует три разных вычисления. В каждом поле применяется выражение, в котором дано более информативное название поля. На рисунке 8.7 результаты отображают одну запись с максимальной ценой, минимальной ценой и средней ценой продуктов, проданных компанией *Boutique Fudge.*

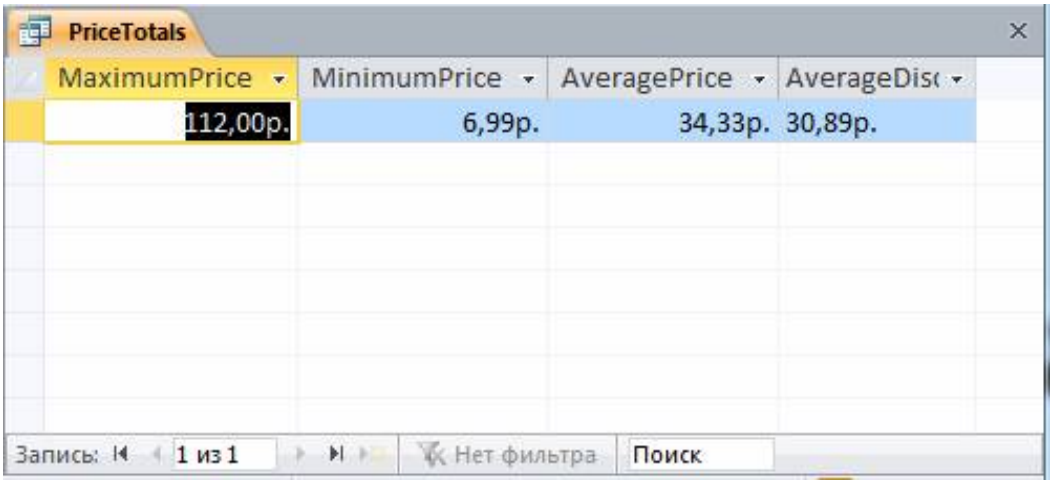

Рис.8.7. Результат создания итогового запроса.

Разрабатывая итоговые запросы, можно использовать все приобретенные ранее навыки написания запросов.

Группировка в итоговом запросе

Простейший итоговый запрос суммирует все выбранные записи в одну строку результатов, как показано на рис. 8.7. В более сложном итоговом запросе применяется группировка для вычисления промежуточных итогов.

Для корректного применения группировки следует помнить о том, что поле, которое используется, должно содержать много повторяющихся значений. На рисунке 8.8 показано создание в режиме конструктора итогового запроса с применением группировки.

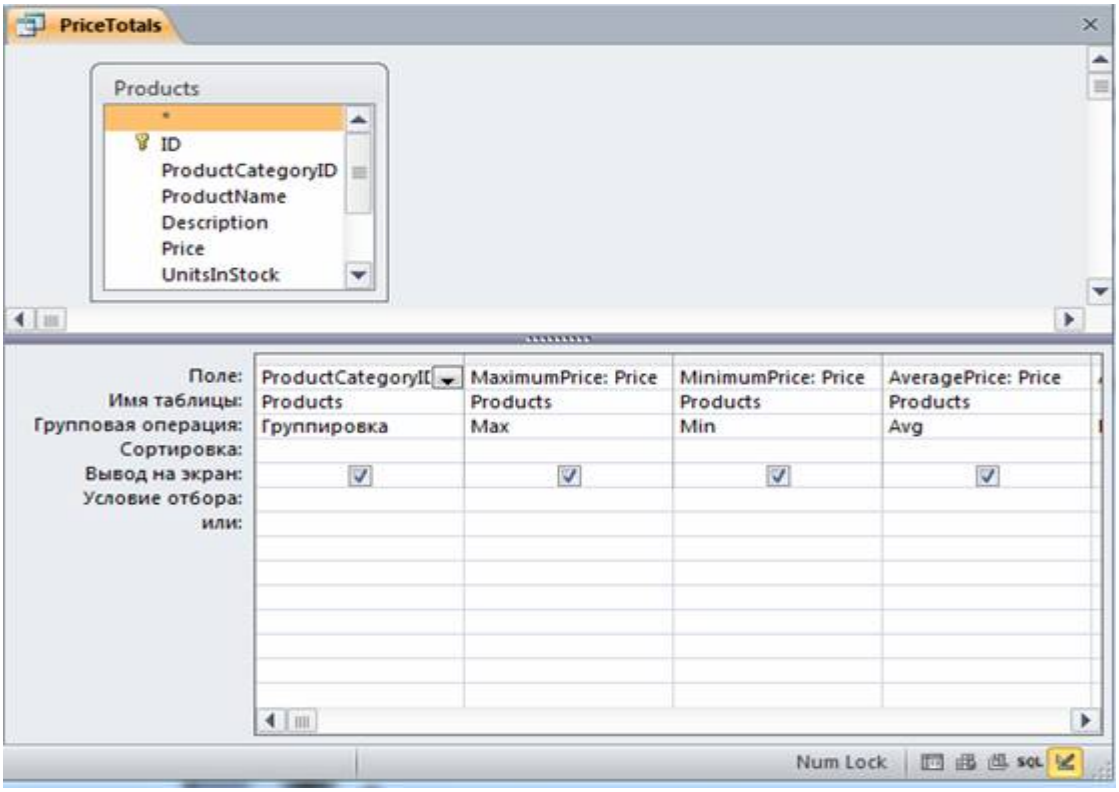

Рис. 8.8. Создание итогового запроса с группировкой

На рисунке 8.9 показан итоговый запрос с применением группировки.

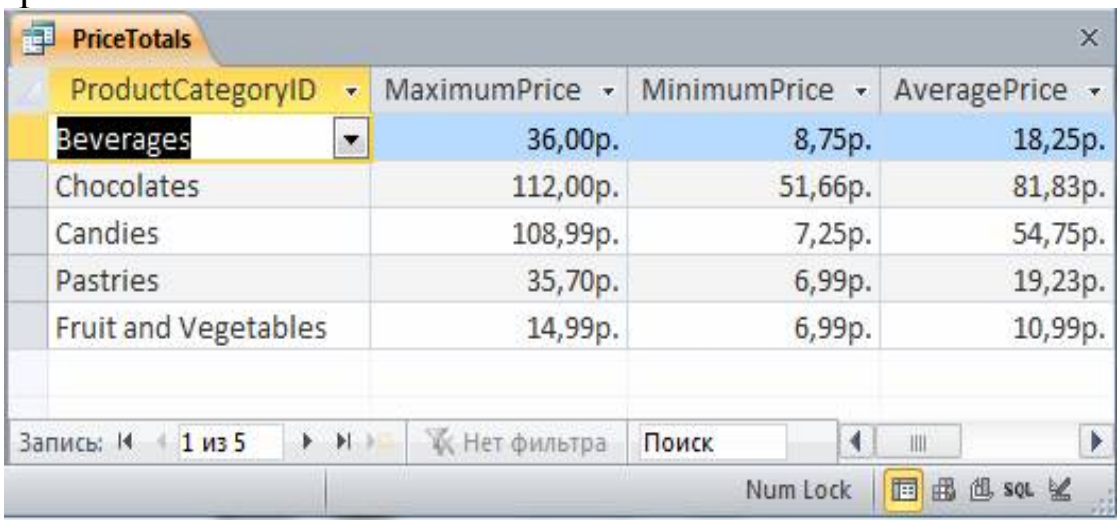

Рис. 8.9. Результат создания итогового запроса

В итоговом запросе можно использовать многоуровневую группировку, вставив несколько полей со значением *Группировка* в ячейку *Групповая операция*. Предположим, что группируется длинный список записей о продажах по наименованиям товаров и по именам клиентов. В результате получится отдельная группа для

каждой комбинации "клиент – товар", в которой группируются записи из таблицы, а затем они сортируются по *коду клиента*.

Если нужен вывод информации на печать, можно создать отчет, включающий многоуровневую группировку и итоги.

Объединения в итоговом запросе

Итоговые запросы невероятно полезны, когда они комбинируются с объединениями для получения связанной информации из нескольких таблиц.

Если вставить в запрос объединение или два, можно извлечь подчиненную информацию из связанных таблиц и добавить ее в результаты. На рис. 8.10 и 8.11 показан пример вычисления общей стоимости каждого заказа. Затем результаты отсортированы по коду клиента *CustomerID*.

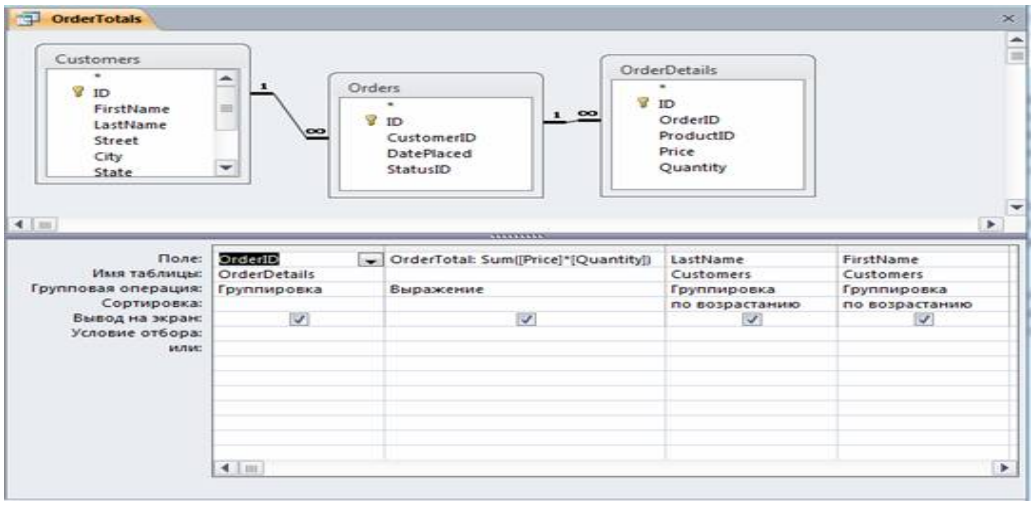

Рис. 8.10. Создание запроса объединения (режим конструктора)

Рис.8.11. Результат создания запроса объединения

Для того, чтобы построить показанный запрос необходимо выполнить следующие действия:

- Создать новый запрос, выбрав *Создание* → *Другие* → *Конструктор запросов*.
- Вставить нужные таблицы с помощью диалогового окна *Добавление таблицы* и затем щелкнуть мышью кнопку *Закрыть*.
- Выбрать *Работа с запросами* | *Конструктор* → *Показать или скрыть* → *Итог.* У каждого поля появится ячейка Групповая операция.
- Добавить поля, которые нужно использовать, и затем в ячейке *Групповая операция* выбрать для каждого поля подходящие группировку или итоговое вычисление. Поля можно выбирать из любых связанных таблиц.
- Теперь можно выполнить запрос.

Параметры запроса

Параметры запроса позволяют создавать сверхгибкие запросы за счет умышленного пропуска одной или нескольких порций информации. При каждом запуске запроса Access запрашивает пропущенные значения. Эти недостающие значения и называют параметрами запроса. Обычно параметры запроса применяют в условиях отбора.

Для создания запроса с параметрами нужно выполнить следующие действия:

- Создать новый запрос, выбрав на ленте *Создание* → *Другие* → *Конструктор запросов.*
- Из диалогового окна *Добавление таблицы* вставить нужные таблицы и щелкнуть мышью кнопку *Закрыть.*
- Выбрать *Работа с запросами* | *Конструктор* → *Показать или скрыть* → *Параметры*. На экране появится диалоговое окно *Параметры запроса*.
- Выбрать имя и тип данных для нужного параметра. Можно использовать любое понравившееся имя (но не применять имя, которое используется для обозначения поля в запросе). Тип данных должен соответствовать типу данных поля, для которого используется параметр. Тип данных задается выбором одного из вариантов в

раскрывающемся списке. Самые распространенные варианты: *Текстовый*, *Целый*, *Денежный* и *Дата/время*.

- Щелкнуть мышью кнопку *ОК* для закрытия окна *Параметры запроса*.

Теперь можно ссылаться на параметр по имени так же, как выполняется ссылка на поле в запросе. Нужно убедиться, что не забыта вставка квадратных скобок, чтобы программа Access знала, что нет попытки ввести фрагмент текста. Во время выполнения запроса Access откроет диалоговое окно *Введите значение параметра* для ввода конкретного значения. Несмотря на то, что такая возможность есть, лучше не применять несколько параметров в одном запросе. Когда запрос выполняется, программа Access отображает отдельное диалоговое окно *Введите значение параметра для каждого значения*. Если применяется несколько параметров, то придется сопровождать выполнение запроса щелчками мышью в этих диалоговых окнах.