**Сортировка результатов запроса с помощью оператора orderby**

Зачастую результаты запроса требуют сортировки. Допустим, что требуется получить

список просроченных счетов по порядку остатка на счету: от самого большого до

самого малого или же список имен заказчиков в алфавитном порядке.

Оператор orderby можно использовать для сортировки результатов запроса по

одному или нескольким критериям. Рассмотрим для начала самый простой случай

сортировки по одному элементу. Ниже приведена общая форма оператора orderby

для сортировки результатов запроса по одному критерию:

orderby элемент порядок

где *элемент* обозначает конкретный элемент, по которому проводится сортировка.

Это может быть весь элемент, хранящийся в источнике данных, или только часть одного

поля в данном элементе. А *порядок* обозначает порядок сортировки по нарастающей

или убывающей с обязательным добавлением ключевого слова ascending или

descending соответственно. По умолчанию сортировка проводится по нарастающей,

и поэтому ключевое слово ascending, как правило, не указывается.

Ниже приведен пример программы, в которой оператор orderby используется

для извлечения значений из массива типа int по нарастающей.

(***glava19\_5***)

class OrderbyDemo

{

static void Main()

{

int[] nums = { 10, -19, 4, 7, 2, -5, 0 };

//make query to get values in order

var posNums = from n in nums

orderby n

select n;

Console.Write("Values ascending:");

//make query and get results

foreach (var i in posNums) Console.Write(i + " ");

Console.WriteLine();

}

}

Для того чтобы изменить порядок сортировки по нарастающей на сортировку по

убывающей, достаточно указать ключевое слово descending, как показано ниже.

var posNums = from n in nums

orderby n descending

select n;

Попробовав выполнить этот запрос, вы получите значения в обратном порядке.

Зачастую сортировка результатов запроса проводится по единственному критерию.

Тем не менее для сортировки по нескольким критериям служит приведенная ниже

форма оператора orderby.

orderby элемент\_А направление, элемент\_В направление, элемент\_С направление, ...

В данной форме *элемент\_А* обозначает конкретный элемент, по которому проводится

основная сортировка; *элемент\_В* — элемент, по которому производится сортировка

каждой группы эквивалентных элементов; *элемент\_С* — элемент, по которому

производится сортировка всех этих групп, и т.д. Таким образом, каждый последующий

*элемент* обозначает дополнительный критерий сортировки. Во всех этих критериях

указывать *направление* сортировки необязательно, но по умолчанию сортировка проводится

по нарастающей. Ниже приведен пример программы, в которой сортировка

информации о банковских счетах осуществляется по трем критериям: фамилии, имени

и остатку на счете.

(***glava19\_6***)

class Account

{

public string FirstName { get; private set; }

public string LastName { get; private set; }

public double Balance { get; private set; }

public string AccountNumber { get; private set; }

public Account(string fn, string ln, string accnum, double b)

{

FirstName = fn;

LastName = ln;

AccountNumber = accnum;

Balance = b;

}

}

class OderbyDemo

{

static void Main()

{

Account[] accounts =

{new Account("Tom", "Smith", "123CK", 100.23),

new Account("Tom", "Smith", "132CD", 10000.00),

new Account("Ralph", "Johns", "436CD", 1923.85),

new Account("Ralph", "Johns", "454MM", 987.132),

new Account("Ted", "Cramer", "897CD", 3223.85),

new Account("Ralph", "Johns", "434CK", -123.32),

new Account("Sarah", "Smith", "547CD", 34955.79),

new Account("Sarah", "Smith", "843CK", 345.00),

new Account("Albert", "Smith", "445CK", -213.67),

new Account("Betty", "Cramer", "968MM", 5146.67),

new Account("Carl", "Smith", "078CD", 15345.99),

new Account("Jenny", "Johns", "108CK", 10.98)

};

//make Query to get data about acc numbers in order

//first order by Names, LastNames, Balance

var accInfo = from acc in accounts

orderby acc.LastName, acc.FirstName, acc.Balance

select acc;

Console.WriteLine("Sorted Acc: ");

string str = "";

foreach (var acc in accInfo)

{

if (str != acc.LastName)

{

Console.WriteLine();

str = acc.LastName;

}

Console.WriteLine("{0}, {1}\tAcc number: {2}, {3,10:C}",

acc.LastName, acc.FirstName, acc.AccountNumber, acc.Balance);

}

Console.WriteLine();

}

}

Внимательно проанализируем оператор orderby в следующем запросе из приведенной

выше программы.

var accInfo = from acc in accounts

orderby acc.LastName, acc.FirstName, acc.Balance

select acc;

Сортировка результатов этого запроса осуществляется следующим образом. Сначала

результаты сортируются по фамилии, затем элементы с одинаковыми фамилиями

сортируются по имени. И наконец, группы элементов с одинаковыми фамилиями и

именами сортируются по остатку на счете.

Используя несколько критериев, можно изменить на обратный порядок любой сортировки

с помощью ключевого слова descending. Например, результаты следующего

запроса будут выведены по убывающей остатков на счетах.

var accInfo = from acc in accounts

orderby acc.LastName, acc.FirstName, acc.Balance descending

select acc;