Оператор select определяет конкретный тип элементов, получаемых по запросу.

Ниже приведена его общая форма.

select acc;

В предыдущих примерах оператор select использовался для возврата переменной

диапазона. Поэтому *выражение* в нем просто обозначало имя переменной диапазона.

Но применение оператора select не ограничивается только этой простой

функцией. Он может также возвращать отдельную часть значения переменной диапазона,

результат выполнения некоторой операции или преобразования переменной

диапазона и даже новый тип объекта, конструируемого из отдельных фрагментов информации,

извлекаемой из переменной диапазона. Такое преобразование исходных

данных называется *проецированием.*

Начнем рассмотрение других возможностей оператора select с приведенной

ниже программы. В этой программе выводятся квадратные корни положительных

значений, содержащихся в массиве типа double.

(***glava19\_7***)

class SelectDemo

{

static void Main()

{

double[] nums =

{-10.0, 16.4, 12.125, 100.85, -2.2, 25.25, -3.5};

//make query to get sqrt from all positive nums int nums

var sqrRoots = from n in nums

where n > 0

select Math.Sqrt(n);

Console.WriteLine("Square roots of positive numbers: ");

foreach (double r in sqrRoots) Console.WriteLine("{0:#.##}", r);

}

}

Обратите особое внимание в данном примере запроса на следующий оператор

select.

select Math.Sqrt(n);

Он возвращает квадратный корень значения переменной диапазона. Для этого значение

переменной диапазона передается методу Math.Sqrt(), который возвращает

квадратный корень своего аргумента. Это означает, что последовательность результатов,

получаемых при выполнении запроса, будет содержать квадратные корни положительных

значений, хранящихся в массиве nums.

Ниже приведена программа, демонстрирующая другое применение оператора

select. В этой программе сначала создается класс EmailAddress, содержащий два

свойства. В первом из них хранится имя адресата, а во втором — адрес его электронной

почты. Затем в этой программе создается массив, содержащий несколько элементов

данных типа EmailAddress. И наконец, в данной программе создается список, состоящий

только из адресов электронной почты, извлекаемых по запросу.

(***glava19\_8***)

class EmailAddress

{

public string Name { get; set; }

public string Address { get; set; }

public EmailAddress(string n, string a)

{

Name = n;

Address = a;

}

}

class SelectDemo2

{

static void Main()

{

EmailAddress[] addrs =

{new EmailAddress("Herbert", "Herb@HerbSchildt.com"),

new EmailAddress("Tom", "Tom@HerbSchieldt.com"),

new EmailAddress("Sarah", "Sara@HerbSchilds.com")};

//get emails

var eAddrs = from entry in addrs

select entry.Address;

Console.WriteLine("Emails: ");

foreach (var i in eAddrs) Console.WriteLine(" " + i);

}

}

Обратите особое внимание на следующий оператор select.

select entry.Address;

Вместо полного значения переменной диапазона этот оператор возвращает лишь

его адресную часть (Address). Это означает, что по данному запросу возвращается

последовательность символьных строк, а не объектов типа EmailAddress. Именно

поэтому переменная s указывается в цикле foreach как string. Ведь как пояснялось

ранее, тип последовательности результатов, возвращаемых по запросу, определяется

типом значения, возвращаемым оператором select.

Одной из самых эффективных для оператора select является возможность возвращать

последовательность результатов, содержащую элементы данных, формируемые

во время выполнения запроса. В качестве примера рассмотрим еще одну программу.

В ней определяется класс ContactInfo, в котором хранится имя, адрес электронной

почты и номер телефона адресата. Кроме того, в этой программе определяется класс

EmailAddress, использовавшийся в предыдущем примере. В методе Main() создается

массив объектов типа ContactInfo, а затем объявляется запрос, в котором источником

данных служит этот массив, но возвращаемая последовательность результатов

содержит объекты типа EmailAddress. Таким образом, типом последовательности

результатов, возвращаемой оператором select, является класс EmailAddress, а не

класс ContactInfo, причем его объекты создаются во время выполнения запроса.

(***glava19\_9***)

class ContactInfo

{

public string Name { get; set; }

public string Email { get; set; }

public string Phone { get; set; }

public ContactInfo(string n, string a, string p)

{

Name = n;

Email = a;

Phone = p;

}

}

class EmailAddress

{

public string Name { get; set; }

public string Address { get; set; }

public EmailAddress(string n, string a)

{

Name = n;

Address = a;

}

}

class SelectDemo3

{

static void Main()

{

ContactInfo[] contacts =

{

new ContactInfo("Herbert", "Herb@HerbSchildt.com", "555-1010"),

new ContactInfo("Tom", "Tom@HerbSchildt.com", "555-1101"),

new ContactInfo("Sara", "Sara@HerbSchildt.com", "555-0110")

};

var eAddrs = from con in contacts

select new EmailAddress(con.Name, con.Email);

foreach (var i in eAddrs) Console.WriteLine(i.Name + ": "+ i.Address);

}

}

Обратите особое внимание в данном запросе на следующий оператор select.

select new EmailAddress(con.Name, con.Email);

В этом операторе создается новый объект типа EmailAddress, содержащий имя

и адрес электронной почты, получаемые из объекта типа ContactInfo, хранящегося

в массиве contacts. Но самое главное, что новые объекты типа EmailAddress создаются

в операторе select во время выполнения запроса.