LINQ 2 Entities

Типы LINQ

	Enumerable	Queryable
Выполнение	В памяти	Удаленно
Реализация	Объекты итераторы	Дерево выражений
Интерфейс	IEnumerable<T>	IQueryable<T>
Провайдеры	LINQ 2 Objects	LINQ 2 SQL LINQ 2 Entities

2

Интерфейс **IEnumerable** находится в пространстве имен **System.Collections**.

- Объект **IEnumerable** представляет **набор данных в памяти** и может перемещаться по этим данным **только вперед**.
- Запрос, представленный объектом **IEnumerable**, **выполняется немедленно** и полностью, поэтому получение данных приложением происходит быстро.
- При выполнении запроса **IEnumerable** загружает **все данные**, и если нам надо выполнить их фильтрацию, то сама фильтрация происходит на стороне клиента.

Интерфейс **IQueryable** располагается в пространстве имен **System.Linq**.

- Объект **IQueryable** предоставляет удаленный доступ к базе данных и позволяет перемещаться по данным как в прямом порядке от начала до конца, так и в обратном порядке.
- В процессе создания запроса, возвращаемым объектом которого является **IQueryable**, происходит оптимизация запроса. В итоге в процессе его выполнения тратится меньше памяти, меньше пропускной способности сети, но в то же время он может обрабатываться чуть медленнее, чем запрос, возвращающий объект **IEnumerable**.

```
IEnumerable<Phone> phoneIEnum = db.Phones;
phoneIEnum = phoneIEnum.Where(p => p.Id > id);
```

```
SELECT
    [Extent1].[Id] AS [Id],
    [Extent1].[Name] AS [Name],
    [Extent1].[Company] AS [Company]
    FROM [dbo].[Phones] AS [Extent1]
```

```
IQueryable<Phone> phoneIQuer = db.Phones;
phoneIQuer = phoneIQuer.Where(p => p.Id > id);
```

```
SELECT
    [Extent1].[Id] AS [Id],
    [Extent1].[Name] AS [Name],
    [Extent1].[Company] AS [Company]
    FROM [dbo].[Phones] AS [Extent1]
    WHERE [Extent1].[Id] >3
```

Для реализации LINQ-провайдеров к внешним по отношению к приложению данным используется интерфейс **IQueryable<T>** (наследник от IEnumerable<T>) вместе с набором методов-расширений, почти полностью идентичных тем, что написаны для IEnumerable<T>

```
public interface IQueryable<out T> :
                                IEnumerable<T>, IEnumerable, IQueryable
{
}
```

Особенность методов-расширений к IQueryable<T> является

- не содержат логики обработки данных (формируют синтаксическую структуру с описанием запроса, «наращивая» ее при каждом новом вызове метода в цепочке)
- при вызове же агрегатных методов или при перечислении описание запроса передается на выполнение провайдеру, инкапсулированному внутри конкретной реализации IQueryable<T>, а тот уже преобразует запрос в язык источника данных, с которым работает, и выполняет его.

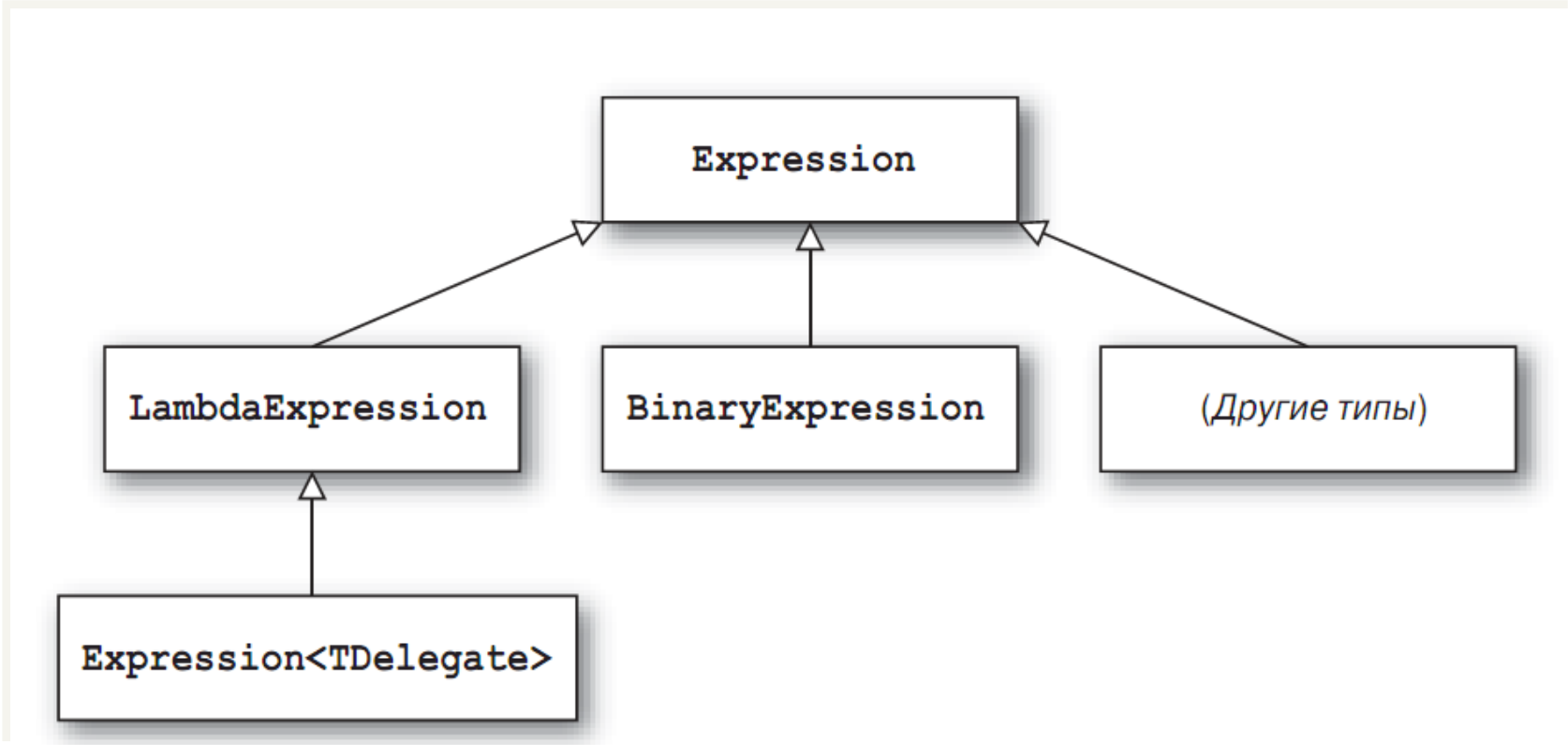
```
public interface IQueryable : IEnumerable
{
  Type ElementType { get; }
  Expression Expression { get; }
  IQueryProvider Provider { get; }
}
```

- для хранения дерева с описанием LINQ-запроса используется свойство Expression
- свойство ElementType содержит информацию о типе возвращаемых запросом элементов и используется в реализациях LINQ-провайдеров для проверки соответствия типов;
- свойство IQueryable.Provider указывает на связанный экземпляр IQueryProvider.

```
public interface IQueryProvider
{
 IQueryable CreateQuery(Expression expression);
 IQueryable<TElement> CreateQuery<TElement>(Expression expression);
 object Execute(Expression expression);
 TResult Execute<TResult>(Expression expression);
}
```

```
public static IQueryable<TSource> Where<TSource>
        (this IQueryable<TSource> source, Expression<Func<TSource, int, bool>> predicate)
{
    if (source == null) throw Error.ArgumentNull("source");
    if (predicate == null) throw Error.ArgumentNull("predicate");
    return source.Provider.CreateQuery<TSource>(
           Expression.Call(
           null,
          ((MethodInfo)MethodBase.GetCurrentMethod()).MakeGenericMethod(typeof(TSource)),
          new Expression[] { source.Expression, Expression.Quote(predicate) }
          ));
      }
```

Деревья выражений представляют собой деревья объектов, в которых каждый узел сам является выражением.



Выполнение запроса LINQ 2 Entities

- query.ToList();
- foreach (var x in query) {...}
- query.First();
- ...

EF выполнит:

1. Преобразование запроса в T-SQL
2. Выполнение запроса на БД
3. Получение результатов
4. Создание объектов из результатов

9

Выполнение запроса LINQ 2 Entities

```
from p in context.People select p
```

Преобразование компилятора

```
IQueryable<Person> people = context.People.Select(p=>p)
```

Выполнение запроса (ToList, First, GetEnumerator,…)

```
SELECT … FROM PEOPLE…
```

Получение и преобразование ответа СУБД в граф объектов

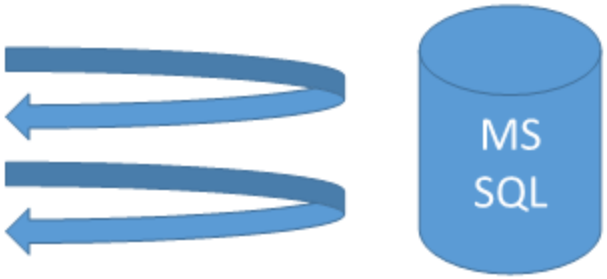
10

Выполнение запроса LINQ 2 Entities

```
IQueryable<Customer> query = context.Customers
                .OrderBy(c => c.LastName)
                .Where(c => c.FirstName == "Robert");

foreach (var customer in query) {}

IEnumerable<Customer> customers = query.ToList();
```




```
int customersCount = customers.Count();

foreach (var customer in customers) {}
```

11

Особенности LINQ 2 Entities

Типы возврата:

- **Enumerable**
- **Queryable**

Методы Linq 2 Entities

- **Where()**
- **Find ()**
- **First () / FirstOrDefault ()**
- **Include ()**
- **Select ()**
- **OrderBy ()**
- **OrderByDescending ()**
- **ThenBy ()/ThenByDescending()**
- **Join ()**
- **GroupBy ()**
- **Union()**
- **Intersect()**
- **Except()**
- **Count ()**
- **Min(),Max()иAverage()**
- **Sum()**

LINQ 2 Entities

```
using (var db = new AWEntities())
{
    IQueryable<Customer> query = from c in db.Customer
                                 select c;

     Console.WriteLine(query);

     foreach (var item in query)
     {
          Console.WriteLine(item);
     }
}
```

```
SELECT
[Extent1].[CustomerID] AS [CustomerID],
[Extent1].[NameStyle] AS [NameStyle],
[Extent1].[Title] AS [Title],
[Extent1].[FirstName] AS [FirstName],
[Extent1].[MiddleName] AS [MiddleName],
[Extent1].[LastName] AS [LastName],
[Extent1].[Suffix] AS [Suffix],
[Extent1].[CompanyName] AS [CompanyName],
[Extent1].[SalesPerson] AS [SalesPerson],
[Extent1].[EmailAddress] AS [EmailAddress],
[Extent1].[Phone] AS [Phone],
[Extent1].[PasswordHash] AS [PasswordHash],
[Extent1].[PasswordSalt] AS [PasswordSalt],
[Extent1].[rowguid] AS [rowguid],
[Extent1].[ModifiedDate] AS [ModifiedDate]
FROM [SalesLT].[Customer] AS [Extent1]
```

LINQ 2 Entities

```
var query = from c in db.Customer
            group c by c.Title into cus
            select new
               { Key = cus.Key, Count = cus.Count(), Group = cus };
```

```
SELECT
[Project1].[C2] AS [C1],
[Project1].[Title] AS [Title],
[Project1].[C1] AS [C2],
[Project1].[C3] AS [C3],
[Project1].[CustomerID] AS [CustomerID],
[Project1].[NameStyle] AS [NameStyle],
[Project1].[Title1] AS [Title1],
[Project1].[FirstName] AS [FirstName],
[Project1].[MiddleName] AS [MiddleName],
[Project1].[LastName] AS [LastName],
[Project1].[Suffix] AS [Suffix],
[Project1].[CompanyName] AS [CompanyName],
[Project1].[SalesPerson] AS [SalesPerson],
[Project1].[EmailAddress] AS [EmailAddress],
[Project1].[Phone] AS [Phone],
[Project1].[PasswordHash] AS [PasswordHash],
[Project1].[PasswordSalt] AS [PasswordSalt],
[Project1].[rowguid] AS [rowguid],
[Project1].[ModifiedDate] AS [ModifiedDate]
FROM ( SELECT
        [GroupBy1].[A1] AS [C1],
        [GroupBy1].[K1] AS [Title],
        1 AS [C2],
        [Extent2].[CustomerID] AS [CustomerID],
        [Extent2].[NameStyle] AS [NameStyle],
        [Extent2].[Title] AS [Title1],
        [Extent2].[FirstName] AS [FirstName],
        [Extent2].[MiddleName] AS [MiddleName],
        [Extent2].[LastName] AS [LastName],
        [Extent2].[Suffix] AS [Suffix],
        [Extent2].[CompanyName] AS [CompanyName],
        [Extent2].[SalesPerson] AS [SalesPerson],
        [Extent2].[EmailAddress] AS [EmailAddress],
        [Extent2].[Phone] AS [Phone],
        [Extent2].[PasswordHash] AS [PasswordHash],
        [Extent2].[PasswordSalt] AS [PasswordSalt],
        [Extent2].[rowguid] AS [rowguid],
        [Extent2].[ModifiedDate] AS [ModifiedDate],
        CASE WHEN (([Extent2].[CustomerID] IS NULL) THEN CAST(NULL AS int) ELSE 1
 END AS [C3]
        FROM   (SELECT
                 [Extent1].[Title] AS [K1],
                 COUNT(1) AS [A1]
                 FROM [SalesLT].[Customer] AS [Extent1]
                 GROUP BY [Extent1].[Title] ) AS [GroupBy1]
        LEFT OUTER JOIN [SalesLT].[Customer] AS [Extent2] ON (([GroupBy1].[K1] =
[Extent2].[Title]) OR ((([GroupBy1].[K1] IS NULL) AND ([Extent2].[Title] IS NULL)
)
)  AS [Project1]
ORDER BY [Project1].[Title] ASC, [Project1].[C3] ASC
```

LINQ 2 Entities

```
using (var context = new AdventureWorksLT2012Entities())
{
    var query = from c in context.Customers select c;
    Console.WriteLine(query);
    Console.ReadKey();
    foreach (var customer in query)
    {
        Console.WriteLine(customer.CustomerID);
    }
    Console.ReadKey();

int customersCount = query.Count();

    Console.WriteLine("Total number of records:{0}", customersCount);
}
```


15

LINQ 2 Entities

```
using (var context = new AdventureWorksLT2012Entities())
{
  IQueryable<Customer> query = from c in context.Customers select c;
  Customer firstCustomer = query.First();

  Customer firstCustomerAll = query.ToList().First();

  Console.WriteLine("First Customer: {0} {1}",
                firstCustomer.FirstName, firstCustomer.LastName);
}
```

LINQ 2 Entities. Вложенные запросы

```
var query = from customer in context.Customers
            orderby customer.FirstName
            select customer;

Console.WriteLine(query);
Console.WriteLine();

var nestedQuery = from a in query
                  where a.CustomerID < 10
                  select a;
Console.WriteLine(nestedQuery);
```


17

LINQ 2 Entities. Ленивая загрузка

```
using (var context = new AdventureWorksLT2012Entities())
{
    context.Configuration.LazyLoadingEnabled = true;
   //Code-First defaults: true, DB/ModelFirst - EDMX Props

    var query = from order in context.SalesOrderHeaders
                select order;
    Console.WriteLine(query);
    var ordersList = query.ToList();

    foreach (var order in ordersList)
    {
        // На каждой итерации обращение к БД!!!!
        if (order.Customer != null)
        Console.WriteLine(order.Customer.LastName);
    }
}
```

LINQ 2 Entities. Строгая загрузка

```
using (var context = new AdventureWorksLT2012Entities())
{
    context.Configuration.LazyLoadingEnabled = false;

    var query = from order in
                context.SalesOrderHeaders.Include("Customer")
                select order;

    var ordersList = query.ToList();
    foreach (var order in ordersList)
    {
        //Нет запросов к БД!!!
        Console.WriteLine(order.Customer.LastName);
    }
}
```

Выполняется Join

LINQ 2 Entities. Явная загрузка

```
using (var context = new AdventureWorksLT2012Entities())
{
    context.Configuration.LazyLoadingEnabled = false;
    context.Customers.Load();

    var query = from order in context.SalesOrderHeaders
                select order;

    Console.WriteLine(query);
    var ordersList = query.ToList();
    foreach (var order in ordersList)
    {
        //Нет запросов к БД!!!
        if (order.Customer != null)
        Console.WriteLine(order.Customer.LastName);
    }
}
```

Загружает все данные в контекст