Декларативно програмиране в процедурните езици

LINQ и заявки

Владислав Ненчев

Софийски Университет “Св. Климент Охридски”
Факултет по Математика и Информатика
Катедра по Математическа Логика и Приложенията ѝ

16 Декември 2016

SQL заявки

```
SELECT
        C.CustomerId,
        FIRST(C.CustomerName) AS CustomerName,
        SUM(S.Amount) AS TotalAmount
FROM Sales AS S
        INNER JOIN Customers AS C
                ON C.CustomerId = S.CustomerId
WHERE C.CorporateCustomer = True
GROUP BY S.CustomerId
HAVING SUM(S.Amount) > 10000
ORDER BY FIRST(C.CustomerName)
```

LINQ заявки

LINQ - нотация за обработка на итерации, наподобяваща SQL.

LINQ заявки

LINQ - нотация за обработка на итерации, наподобяваща SQL.

Превежда се до IEnumerable.Where, IEnumerable.Select, IEnumerable.OrderBy, IEnumerable.GroupBy, IEnumerable.First, IEnumerable.Min, IEnumerable.Average и др.

LINQ заявки

LINQ - нотация за обработка на итерации, наподобяваща SQL.

Превежда се до IEnumerable.Where, IEnumerable.Select, IEnumerable.OrderBy, IEnumerable.GroupBy, IEnumerable.First, IEnumerable.Min, IEnumerable.Average и др.

- LINQ $\rightarrow$ обекти (C# и други .NET езици)
- LINQ $\rightarrow$ SQL (база данни)
- LINQ $\rightarrow$ XML (XML документ)
- LINQ $\rightarrow$ DataSet (ADO.NET dataset)
- LINQ $\rightarrow$ Entity (Microsoft Entity Framework)
- LINQ $\rightarrow$ други (чрез IEnumerable и IQueryable)

Постановка

```
class Sale
{
  public int SaleId;
  public DateTime Date;
  public int CustomerId;
  public decimal Amount;
}

class Customer
{
  public int CustomerId;
  public string CustomerName;
  public bool CorporateCustomer;
}
```

Основна структура на LINQ заявките

```
var query =
  from sale in sales
  where sale.Date.Year == 2016
  select sale.SaleId;

query.ForEach(saleId => Console.WriteLine("Id of a
    sale within year 2016 is: {0}.", saleId));
```

Основна структура на LINQ заявките

```
var query =
  from sale in sales
  where sale.Date.Year == 2016
  select sale.SaleId;

query.ForEach(saleId => Console.WriteLine("Id of a
    sale within year 2016 is: {0}.", saleId));
```

Резултата query е IEnumerable<int>.

sales е или IEnumerable или IQueryable.

where не е задължително.

Заявката се изпълнява чак при ForEach.

Източника на данни - from частта

Може да се използват повече от един източници/колекции.

```
public Graph Total
{
  get
  {
    Graph total = new Graph(Name + " (total)");
    total.vertices = vertices;
    total.edges =
      new HashSet<GraphEdge>(
        from vertex1 in vertices
        from vertex2 in vertices
        select Edge(vertex1, vertex2));
    return total;
  }
}
```

Извличане на данните - select частта

```
var query =
  from sale in sales
  where sale.Amount > 2000
  select new { sale.SaleId, sale.Date };
```

Превежда се до:

```
sales.Where(sale => sale.Amount > 2000).
  Select(sale => new { sale.SaleId, sale.Date });
```

Филтриране на данните - where частта

```
var query =
  from sale in sales
  where sale.Date.Year == 2016
  where sale.Amount > 2000
  select sale.SaleId;
```

Превежда се до:

```
sales.Where(sale => sale.Date.Year == 2016).
  Where(sale => sale.Amount > 2000).
    Select(sale => sale.SaleId);
```

Свързване на няколко източника - join

```
var query =
  from sale in sales
    join customer in customers
      on sale.CustomerId equals customer.CustomerId
  select new { sale.SaleId,
    customer.CustomerId, customer.CustomerName };
```

Превежда се до:

```
sales.Join(customers,
  sale => sale.CustomerId,
  customer => customer.CustomerId,
  (sale, customer) =>
    new { sale.SaleId,
      customer.CustomerId, customer.CustomerName });
```

Подредба на данните - orderby

```
var query =
  from sale in sales
  orderby sale.Date, sale.Amount descending
  select new { sale.SaleId,
    sale.Date, sale.Amount };
```

Превежда се до:

```
sales.OrderBy(sale => sale.Date).
  ThenByDescending(sale => sale.Amount).
    Select(sale => new { sale.SaleId,
      sale.Date, sale.Amount });
```

Групиране на данните - group by

```
var query =
  from sale in sales
  group sale by sale.CustomerId into total
  select new { CustomerId = total.Key,
    TotalAmount = total.Sum(s => s.Amount) };
```

Превежда се до:

```
sales.GroupBy(sale => sale.CustomerId).
  Select(total => new { CustomerId = total.Key,
    TotalAmount = total.Sum(s => s.Amount) });
```

Агрегатни функции и филтриране по тях

```
var query =
  from sale in sales
    join customer in customers
      on sale.CustomerId equals customer.CustomerId
  where customer.CorporateCustomer
  group new { customer.CustomerId,
    customer.CustomerName, sale.Amount }
      by customer.CustomerId into total
  where total.Sum(x => x.Amount) > 10000
  select new { CustomerId = total.Key,
    CustomerName =
      total.Select(x => x.CustomerName).First(),
    TotalAmount = total.Sum(x => x.Amount) };
```

Финално решение

```
var query =
  from sale in sales
    join customer in customers
      on sale.CustomerId equals customer.CustomerId
  where customer.CorporateCustomer
  group new { customer.CustomerId,
    customer.CustomerName, sale.Amount }
      by customer.CustomerId into total
  select new { CustomerId = total.Key,
      CustomerName =
        total.Select(x => x.CustomerName).First(),
      TotalAmount = total.Sum(x => x.Amount) };
    into TotalData
    where TotalData.TotalAmount > 10000
    orderby TotalData.CustomerName
    select TotalData;
```

Първоначалната заявка

```
SELECT
        C.CustomerId,
        FIRST(C.CustomerName) AS CustomerName,
        SUM(S.Amount) AS TotalAmount
FROM Sales AS S
        INNER JOIN Customers AS C
                ON C.CustomerId = S.CustomerId
WHERE C.CorporateCustomer = True
GROUP BY S.CustomerId
HAVING SUM(S.Amount) > 10000
ORDER BY FIRST(C.CustomerName)
```

Други оператори в LINQ

- IEnumerable.SelectMany, IEnumerable. Distinct , IEnumerable.OfType, IEnumerable.GroupJoin, IEnumerable.Reverse

Други оператори в LINQ

- IEnumerable.SelectMany, IEnumerable.Distinct, IEnumerable.OfType, IEnumerable.GroupJoin, IEnumerable.Reverse
- IEnumerable.Average, IEnumerable.Count, IEnumerable.FirstOrDefault, IEnumerable.Last, IEnumerable.LastOrDefault, IEnumerable.Single, IEnumerable.Min, IEnumerable.Max

Други оператори в LINQ

- IEnumerable.SelectMany, IEnumerable.Distinct, IEnumerable.OfType, IEnumerable.GroupJoin, IEnumerable.Reverse
- IEnumerable.Average, IEnumerable.Count, IEnumerable.FirstOrDefault, IEnumerable.Last, IEnumerable.LastOrDefault, IEnumerable.Single, IEnumerable.Min, IEnumerable.Max
- IEnumerable.Empty, IEnumerable.Range, IEnumerable.Repeat

Други оператори в LINQ

- IEnumerable.SelectMany, IEnumerable.Distinct, IEnumerable.OfType, IEnumerable.GroupJoin, IEnumerable.Reverse
- IEnumerable.Average, IEnumerable.Count, IEnumerable.FirstOrDefault, IEnumerable.Last, IEnumerable.LastOrDefault, IEnumerable.Single, IEnumerable.Min, IEnumerable.Max
- IEnumerable.Empty, IEnumerable.Range, IEnumerable.Repeat
- IEnumerable.Union, IEnumerable.Intersect, IEnumerable.Except

Други оператори в LINQ

- IEnumerable.SelectMany, IEnumerable. Distinct , IEnumerable.OfType, IEnumerable.GroupJoin, IEnumerable.Reverse
- IEnumerable.Average, IEnumerable.Count, IEnumerable. FirstOrDefault , IEnumerable.Last, IEnumerable.LastOrDefault, IEnumerable. Single , IEnumerable.Min, IEnumerable.Max
- IEnumerable.Empty, IEnumerable.Range, IEnumerable.Repeat
- IEnumerable.Union, IEnumerable. Intersect , IEnumerable.Except
- IEnumerable.Skip, IEnumerable.SkipWhile, IEnumerable.Take, IEnumerable.TakeWhile

Други оператори в LINQ

- IEnumerable.SelectMany, IEnumerable.Distinct, IEnumerable.OfType, IEnumerable.GroupJoin, IEnumerable.Reverse
- IEnumerable.Average, IEnumerable.Count, IEnumerable.FirstOrDefault, IEnumerable.Last, IEnumerable.LastOrDefault, IEnumerable.Single, IEnumerable.Min, IEnumerable.Max
- IEnumerable.Empty, IEnumerable.Range, IEnumerable.Repeat
- IEnumerable.Union, IEnumerable.Intersect, IEnumerable.Except
- IEnumerable.Skip, IEnumerable.SkipWhile, IEnumerable.Take, IEnumerable.TakeWhile
- IEnumerable.AsEnumerable, IEnumerable.AsQueryable, IEnumerable.Cast, IEnumerable.ToArray, IEnumerable.ToDictionary, IEnumerable.ToList, IEnumerable.ToLookup