

第32期-在远程查询数据

2019年6月3日 20:07

生成操作

DefaultIfEmpty、Empty、Range 和 Repeat, IQueryable 只支持 DefaultIfEmpty 方法。

```
var blogs = _context.Blogs.Where(b => b.BlogId > 1).DefaultIfEmpty();
```

```
SELECT [t].[BlogId], [t].[Name], [t].[OwnerId], [t].[Url]
FROM (
    SELECT NULL AS [empty]
) AS [empty]
LEFT JOIN (
    SELECT [b].[BlogId], [b].[Name], [b].[OwnerId], [b].[Url]
    FROM [Blogs] AS [b]
    WHERE [b].[BlogId] > 1
) AS [t] ON 1 = 1
```

```
var defaultBlog = new Blog { Name = "零度" };
var blogs = _context.Blogs.Where(b => b.BlogId > 1).DefaultIfEmpty(defaultBlog);
```

```
SELECT [b].[BlogId], [b].[Name], [b].[OwnerId], [b].[Url]
FROM [Blogs] AS [b]
WHERE [b].[BlogId] > 1
```

```
warn: Microsoft.EntityFrameworkCore.Query[20500]
      The LINQ expression 'DefaultIfEmpty(__p_0)' could not be
translated and will be evaluated locally.
```

相等比较

SequenceEqual

```
var blogs1 = _context.Blogs.Where(b => b.BlogId > 5);
var blogs2 = _context.Blogs.Where(b => b.BlogId < 2);
```

```
var result = blogs1.SequenceEqual(blogs2);
```

未经处理的异常

System.NotSupportedException: "Could not parse expression 'value (Microsoft.EntityFrameworkCore.Query.Internal.EntityQueryable`1 [EntityFrameworkCoreSample.Models.Blog]).Where(b => (b.BlogId > 5)).SequenceEqual (value (Microsoft.EntityFrameworkCore.Query.Internal.EntityQueryable`1 [EntityFrameworkCoreSample.Models.Blog]).Where(b => (b.BlogId < 2)))': This overload of the method 'System.Linq.Queryable.SequenceEqual' is currently not supported."

[查看详细信息](#) | [复制详细信息](#)

异常设置

```
var blogs1 = _context.Blogs.Where(b => b.BlogId > 5).AsEnumerable();
var blogs2 = _context.Blogs.Where(b => b.BlogId < 2).AsEnumerable();

var result = blogs1.SequenceEqual(blogs2);
```

```
SELECT [b].[BlogId], [b].[Name], [b].[OwnerId], [b].[Url]
FROM [Blogs] AS [b]
WHERE [b].[BlogId] > 5
```

```
SELECT [b0].[BlogId], [b0].[Name], [b0].[OwnerId], [b0].[Url]
FROM [Blogs] AS [b0]
WHERE [b0].[BlogId] < 2
```

串联运算

Concat

```
var blogs1 = _context.Blogs.Where(b => b.BlogId > 5);
var blogs2 = _context.Blogs.Where(b => b.BlogId < 2);

var blogs3 = blogs1.Concat(blogs2);
```

```
SELECT [b].[BlogId], [b].[Name], [b].[OwnerId], [b].[Url]
FROM [Blogs] AS [b]
WHERE [b].[BlogId] > 5
```

```
SELECT [b0].[BlogId], [b0].[Name], [b0].[OwnerId], [b0].[Url]
FROM [Blogs] AS [b0]
WHERE [b0].[BlogId] < 2
```

```
warn: Microsoft.EntityFrameworkCore.Query[20500]
      The LINQ expression 'Concat({from Blog b in value(Microsoft.EntityFrameworkCore.Query.Internal.EntityQueryable`1[EntityFrameworkCoreSample.Models.Blog]) where ([b].BlogId < 2) select [b]})' could not be translated and will be evaluated locally.
```

联接运算

Join 和 GroupJoin

```
var query = _context.Blogs.Join(_context.Posts,
    b => b.BlogId,
    p => p.BlogId,
    (b, p) => new { b.BlogId, BlogTitle = p.Title, p.Content }
);
```

```
SELECT [b].[BlogId], [p].[Title] AS [BlogTitle], [p].[Content]
FROM [Blogs] AS [b]
INNER JOIN [Posts] AS [p] ON [b].[BlogId] = [p].[BlogId]
```

```
var query = _context.Blogs.GroupJoin(_context.Posts,
    b => b.BlogId,
    p => p.BlogId,
    (b, ps) => new { b.BlogId, Posts=ps }
);
```

```
SELECT [b].[BlogId] AS [BlogId0], [b].[Name], [b].[OwnerId],
[b].[Url], [p].[PostId], [p].[BlogId], [p].[Content], [p].[Title]
FROM [Blogs] AS [b]
LEFT JOIN [Posts] AS [p] ON [b].[BlogId] = [p].[BlogId]
ORDER BY [BlogId0]
```

数据分组

GroupBy 和 ToLookup

```
var query = _context.Blogs.Where(b=>b.BlogId>2).GroupBy(b=>b.OwnerId);
```

```
SELECT [b].[BlogId], [b].[Name], [b].[OwnerId], [b].[Url]
FROM [Blogs] AS [b]
WHERE [b].[BlogId] > 2
ORDER BY [b].[OwnerId]
```

```
var query = _context.Blogs.Where(b=>b.BlogId>2).GroupBy(b=>b.Owner);
```

```
SELECT [b].[BlogId], [b].[Name], [b].[OwnerId], [b].[Url], [b.Owner
].[PersonId], [b.Owner].[Name]
FROM [Blogs] AS [b]
INNER JOIN [Person] AS [b.Owner] ON [b].[OwnerId] = [b.Owner].[Pers
onId]
WHERE [b].[BlogId] > 2
ORDER BY [b.Owner].[PersonId]
```

```
warn: Microsoft.EntityFrameworkCore.Query[20500]
      The LINQ expression 'GroupBy([b.Owner], [b])' could not be translat
ed and will be evaluated locally.
```

```
var query1 = _context.Blogs.Where(b=>b.BlogId>2).ToLookup(b =>b.OwnerId);
var query2 = _context.Blogs.Where(b => b.BlogId > 2).ToLookup(b => b.Owner);
```

```
SELECT [b].[BlogId], [b].[Name], [b].[OwnerId], [b].[Url]
FROM [Blogs] AS [b]
WHERE [b].[BlogId] > 2
```

投影运算

Select 与 SelectMany

```
var query = _context.Blogs.Select(b => new { BlogName = b.Name + "$", b.Url });
```

```
SELECT [b].[Name] + N'$' AS [BlogName], [b].[Url]
FROM [Blogs] AS [b]
```

```
var query = _context.Blogs
    .Select(b => new { b.BlogId, b.Name, Posts = _context.Posts.Where(p => p.BlogId == b.BlogId) })
    .SelectMany(b => b.Posts);
```

```
SELECT [p].[PostId], [p].[BlogId], [p].[Content], [p].[Title]
FROM [Blogs] AS [b]
CROSS JOIN [Posts] AS [p]
WHERE [p].[BlogId] = [b].[BlogId]
```

[图解各种JOIN的区别](#)

[深入理解SQL中的Join机制](#)

[SQL夯实基础各种联接查询](#)

数据分区

Skip、SkipWhile、Take 和 TakeWhile, SkipLast 和 TakeLast

```
var query = _context.Blogs.Where(b=>b.BlogId>2).Skip(2);
```

```
SELECT [b].[BlogId], [b].[Name], [b].[OwnerId], [b].[Url]
FROM [Blogs] AS [b]
WHERE [b].[BlogId] > 2
ORDER BY (SELECT 1)
OFFSET @__p_0 ROWS
```

[分页实现: OFFSET-FETCH](#)

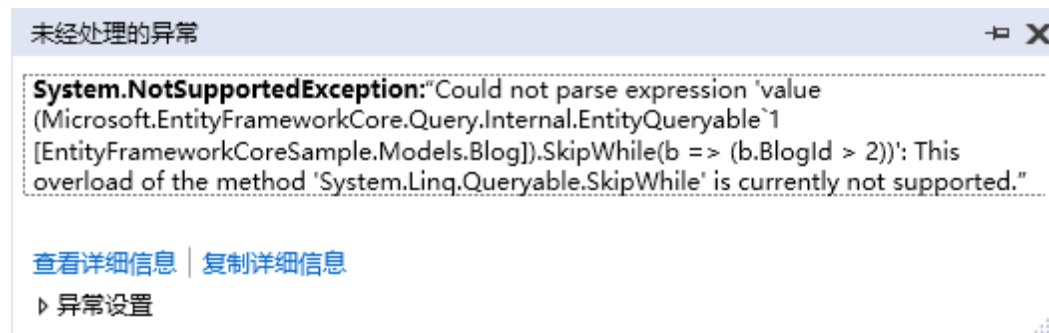
```
var query = _context.Blogs.Where(b=>b.BlogId>2).Take(2);
```

```
SELECT TOP(@__p_0) [b].[BlogId], [b].[Name], [b].[OwnerId], [b].[Url]
FROM [Blogs] AS [b]
WHERE [b].[BlogId] > 2
```

```
var query = _context.Blogs.Where(b => b.BlogId > 2).Skip(10).Take(10);
```

```
SELECT [b].[BlogId], [b].[Name], [b].[OwnerId], [b].[Url]
FROM [Blogs] AS [b]
WHERE [b].[BlogId] > 2
ORDER BY (SELECT 1)
OFFSET @__p_0 ROWS FETCH NEXT @__p_0 ROWS ONLY
```

```
var query = _context.Blogs.SkipWhile(b => b.BlogId > 2);
var query = _context.Blogs.TakeWhile(b => b.BlogId > 2);
var query = _context.Blogs.SkipLast(2);
var query = _context.Blogs.TakeLast(2);
```



数据排序

OrderBy、OrderByDescending、ThenBy、ThenByDescending 和 Reverse

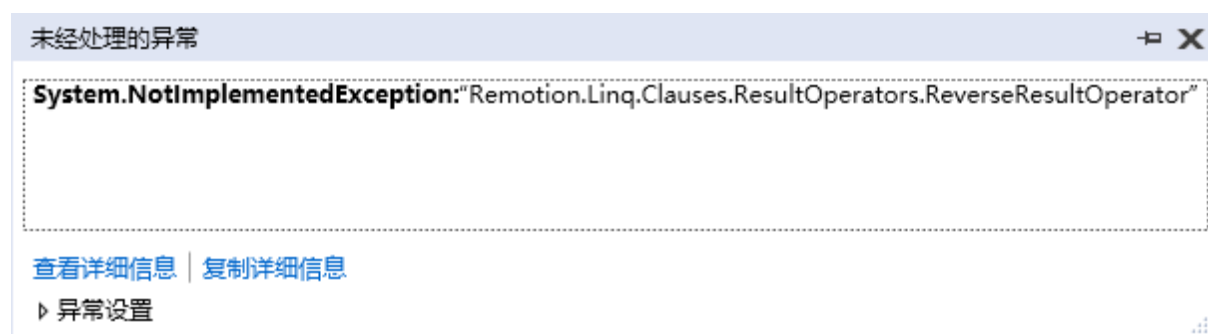
```
var query = _context.Blogs.Where(b => b.BlogId > 2).OrderBy(b => b.OwnerId);
```

```
SELECT [b].[BlogId], [b].[Name], [b].[OwnerId], [b].[Url]
FROM [Blogs] AS [b]
WHERE [b].[BlogId] > 2
ORDER BY [b].[OwnerId]
```

```
var query = _context.Blogs.Where(b => b.BlogId > 2).OrderBy(b =>
b.OwnerId).OrderByDescending(b=>b.BlogId);
var query = _context.Blogs.Where(b => b.BlogId > 2).OrderBy(b => b.OwnerId).ThenByDescending(b
=> b.BlogId);
```

```
SELECT [b].[BlogId], [b].[Name], [b].[OwnerId], [b].[Url]
FROM [Blogs] AS [b]
WHERE [b].[BlogId] > 2
ORDER BY [b].[BlogId] DESC, [b].[OwnerId]
```

```
var query = _context.Blogs.Where(b => b.BlogId > 2).Reverse();
```



```
var query = _context.Blogs.Where(b => b.BlogId > 2).AsEnumerable().Reverse();
```

```
SELECT [b].[BlogId], [b].[Name], [b].[OwnerId], [b].[Url]
FROM [Blogs] AS [b]
WHERE [b].[BlogId] > 2
```

集合运算

Distinct、Except、Intersect 和 Union

```
var blogs = _context.Blogs.Where(b => b.BlogId > 5);  
var query = blogs.Distinct();
```

```
SELECT DISTINCT [b].[BlogId], [b].[Name], [b].[OwnerId], [b].[Url]  
FROM [Blogs] AS [b]  
WHERE [b].[BlogId] > 5
```

```
var blogs1 = _context.Blogs.Where(b => b.BlogId > 5);  
var blogs2 = _context.Blogs.Where(b => b.BlogId > 2);
```

```
var query = blogs1.Except(blogs2);  
var query = blogs1.Intersect(blogs2);  
var query = blogs1.Union(blogs2);
```

```
SELECT [b0].[BlogId], [b0].[Name], [b0].[OwnerId], [b0].[Url]  
FROM [Blogs] AS [b0]  
WHERE [b0].[BlogId] > 2
```

```
SELECT [b].[BlogId], [b].[Name], [b].[OwnerId], [b].[Url]  
FROM [Blogs] AS [b]  
WHERE [b].[BlogId] > 5
```

```
warn: Microsoft.EntityFrameworkCore.Query[20500]  
      The LINQ expression 'Except({from Blog b in value(Microsoft.EntityFrameworkCore.Query.Internal.EntityQueryable`1[EntityFrameworkCoreSample.Models.Blog]) where ([b].BlogId > 2) select [b]})' could not be translated and will be evaluated locally.
```

限定符运算

All、Any 和 Contains

```
bool query = _context.Blogs.All(b => b.BlogId > 5);
```

```
SELECT CASE  
  WHEN NOT EXISTS (  
    SELECT 1  
    FROM [Blogs] AS [b]  
    WHERE [b].[BlogId] <= 5)  
  THEN CAST(1 AS BIT) ELSE CAST(0 AS BIT)  
END
```

```
bool query = _context.Blogs.Any(b => b.BlogId > 5);
```

```

SELECT CASE
    WHEN EXISTS (
        SELECT 1
        FROM [Blogs] AS [b]
        WHERE [b].[BlogId] > 5)
    THEN CAST(1 AS BIT) ELSE CAST(0 AS BIT)
END

```

```

var names = new string[] { "A", "B", "C" };
bool query = _context.Blogs.Any(b => names.Contains(b.Name));

```

```

SELECT CASE
    WHEN EXISTS (
        SELECT 1
        FROM [Blogs] AS [b]
        WHERE [b].[Name] IN (N'A', N'B', N'C'))
    THEN CAST(1 AS BIT) ELSE CAST(0 AS BIT)
END

```

```

var blog = new Blog { BlogId = 1 };
bool query = _context.Blogs.Contains(blog);

```

```

SELECT CASE
    WHEN @_p_0_BlogId IN (
        SELECT [b].[BlogId]
        FROM [Blogs] AS [b]
    )
    THEN CAST(1 AS BIT) ELSE CAST(0 AS BIT)
END

```

```

bool query = _context.Blogs.Where(b => b.BlogId > 2).Contains(blog);

```

```

SELECT CASE
    WHEN @_p_0_BlogId IN (
        SELECT [b].[BlogId]
        FROM [Blogs] AS [b]
        WHERE [b].[BlogId] > 2
    )
    THEN CAST(1 AS BIT) ELSE CAST(0 AS BIT)
END

```

```

var query = _context.Blogs.Where(b => b.Name.Contains("ABC"));

```

```

SELECT [b].[BlogId], [b].[Name], [b].[OwnerId], [b].[Url]
FROM [Blogs] AS [b]
WHERE CHARINDEX(N'ABC', [b].[Name]) > 0

```

```

var query = _context.Blogs.Where(b => b.Name.StartsWith("A"));

```

```

SELECT [b].[BlogId], [b].[Name], [b].[OwnerId], [b].[Url]
FROM [Blogs] AS [b]
WHERE [b].[Name] LIKE N'A' + N'%' AND (LEFT([b].[Name], LEN(N'A')) = N'A')

```

```

var query = _context.Blogs.Where(b => b.Name.EndsWith("ABC"));

```

```

SELECT [b].[BlogId], [b].[Name], [b].[OwnerId], [b].[Url]
FROM [Blogs] AS [b]
WHERE RIGHT([b].[Name], LEN(N'ABC')) = N'ABC'

```

