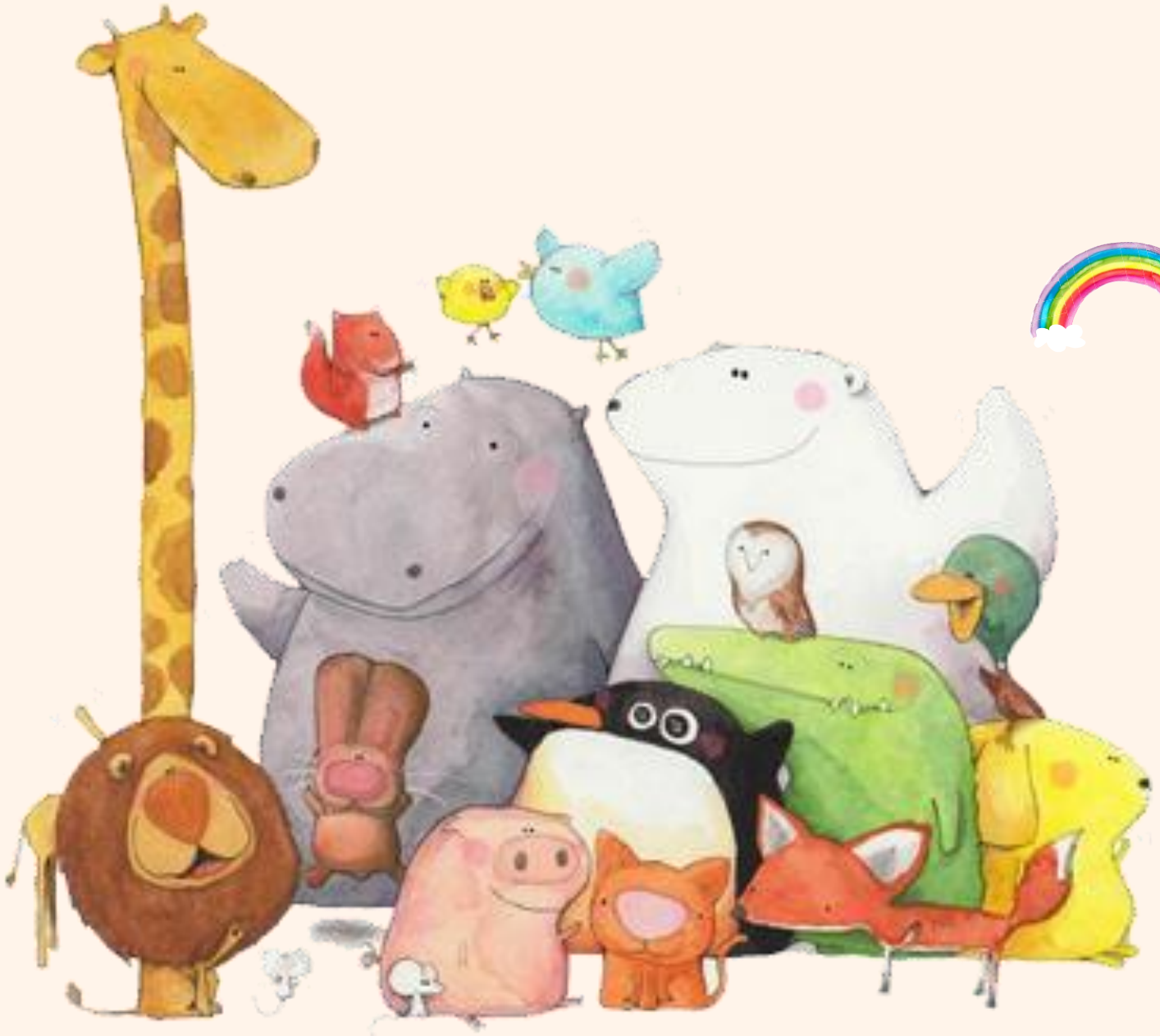


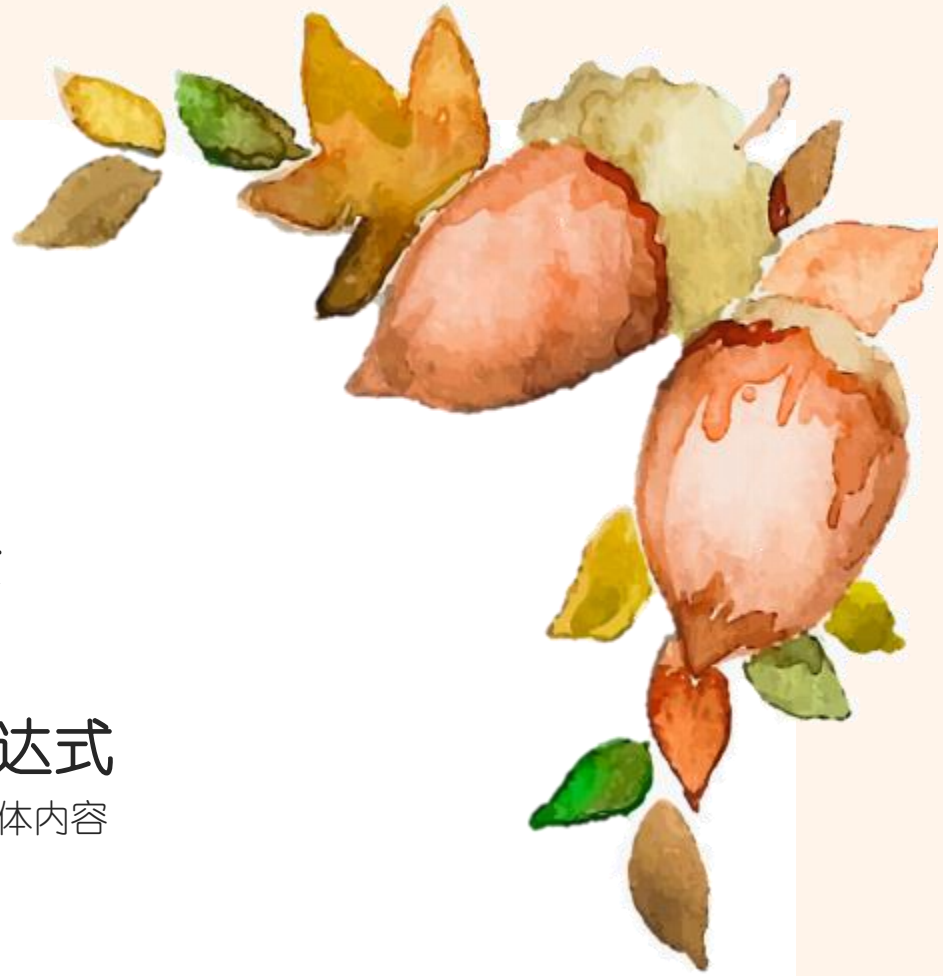
LINQ

厦门大学信息学院 赵江声

2023-10

目录

Content



01

LINQ基本概念

简要介绍

1.1 LINQ概述

参考资料：https://learn.microsoft.com/zh-cn/dotnet/csharp/linq/

语言集成查询（LINQ）是一系列直接将查询功能集成到 C# 语言的技术统称。

- 完整的操作包括创建数据源、定义查询表达式和在 foreach 语句中执行查询。

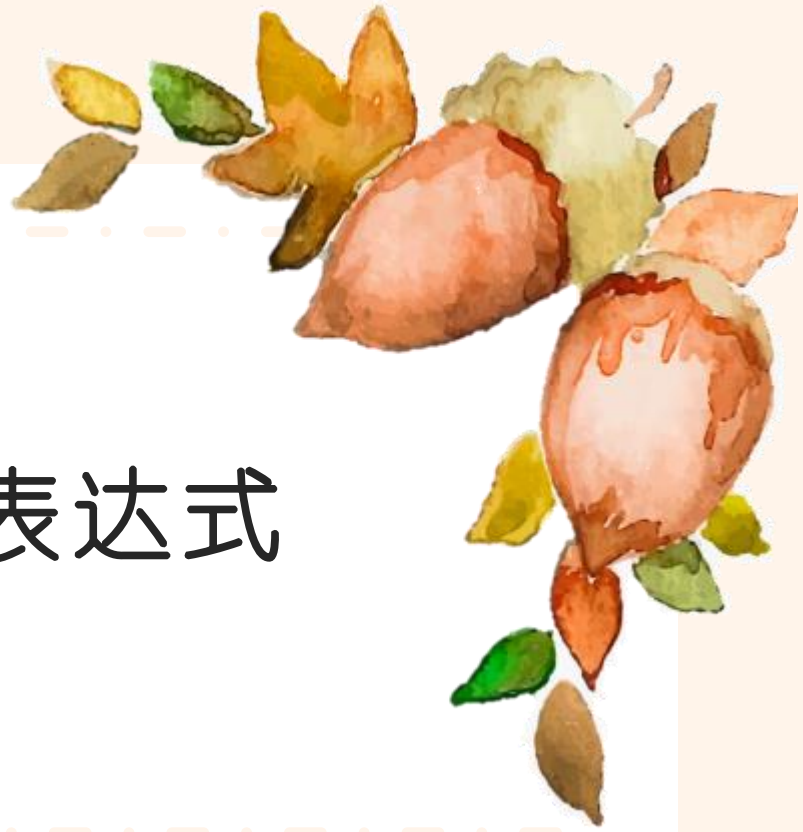
```
int[] scores = { 96, 64, 98, 81, 60 };
// Define the query expression.
IEnumerable<int> scoreQuery =
    from score in scores
    where score > 80
    select score;
/* 上面的查询表达式，用自然语言描述就是:
* “对于定义域scores中的元素，只要符合条件（>80），就选择它”
* 类似于数学的集合 {x | x∈S, x > 80}
* 或者{score|score∈scores, score > 80}
*/
```

```
// Execute the query.
foreach (int i in scoreQuery)
{
    Console.Write(i + ", ");
}
Console.WriteLine();
// Output: 96, 98, 81,

//建议用这种方法来写
Console.WriteLine(string.Join(", ", scoreQuery));
// Output: 96, 98, 81
```

```
//查询表达式还有一个等价的写法
IEnumerable<int> scoreQuery1 =
    scores.Where(score⇒score>80).Select(score ⇒ score);
Console.WriteLine(string.Join(", ", scoreQuery1));
```

1.2 LINQ 与 普通数据查询 的区别

普通数据查询	LINQ
简单的字符串，	查询表达式，
没有编译时类型检查或 IntelliSense 支持	有编译时类型检查或 IntelliSense 支持
无查询语法	采用声明性查询语法，可以用最少的代码对数据源执行筛选、排序和分组操作。
针对每种类型的数据源，需要了解不同的查询语言：SQL 数据库、XML 文档、各种 Web 服务等。	可使用相同的基本查询表达式模式来查询和转换 SQL 数据库、ADO .NET 数据集、XML 文档和流以及 .NET 集合中的数据。



02

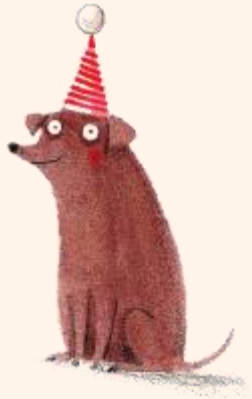
LINQ查询表达式

2.1 基本概念

- 查询表达式是以查询语法表示的查询。

- 查询的三种操作
  - 检索元素的子集以生成新序列，而不修改各个元素。 查询然后可能以各种方式对返回的序列进行排序或分组；
  - 如前面的示例所示检索元素的序列，但是将它们转换为新类型的对象；
  - 检索有关源数据的单独值。

- 查询变量
  - 在 LINQ 中，查询变量是存储查询而不是查询结果的任何变量。
  - 在前边的样例1中，scoreQuery就是查询变量。是可枚举类型。

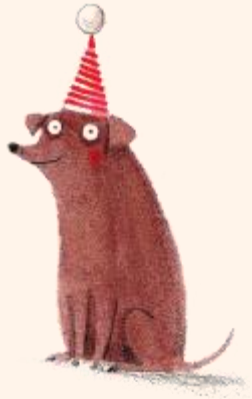


2.2 开头和结尾

查询表达式必须以 from 子句开头，且必须以 select 或 group 子句结尾。

- 开头
  - 必须以 from 子句开头，它指定数据源以及范围变量。
  - 查询表达式可以有多个 from 子句
- 结尾
  - 必须以 group 子句或 select 子句结尾
    - group 子句 ： 使用 group 子句可生成按指定键组织的组的序列。
    - select 子句 ： 使用 select 子句可生成所有其他类型的序列，还可以用于将源数据转换为新类型的序列。

```
IEnumerable<City> cityQuery =
    from country in countries
    from city in country.Cities
    where city.Population > 10000
    select city;
```

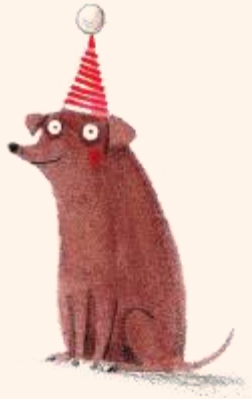


2.3 筛选、排序和联接

在第一个 from 子句与最后一个 select 或 group 子句之间，可以包含以下这些可选子句中的一个或多个：where、orderby、join、let，甚至是其他 from 子句。

还可以使用 into 关键字，使 join 或 group 子句的结果可以充当相同查询表达式中的其他查询子句的源。

```
//数据（本节示例所用数据）
//Scores: 语文，数学，英语，体育
List<Student> students = new List<Student>
{
  new Student {SurName="张", GivenName="三", ID=111, Scores= new List<int>
            {98, 81, 92, 72}},
  new Student {SurName="李", GivenName="四", ID=112, Scores= new List<int>
            {65, 88, 91, 39}},
  new Student {SurName="王", GivenName="五", ID=113, Scores= new List<int>
            {88, 84, 75, 91}},
  new Student {SurName="赵", GivenName="六", ID=114, Scores= new List<int>
            {97, 89, 85, 82}},
```



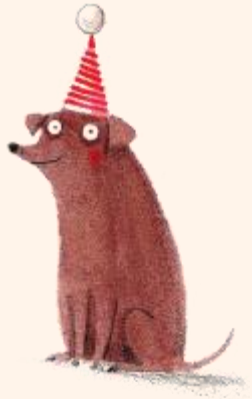
2.3.1 where 子句

可基于一个或多个谓词表达式，从源数据中筛选出元素。

【例】从上述数据中选出语文成绩>=90 并且 总成绩>=平均值 的所有学生;

复制代码

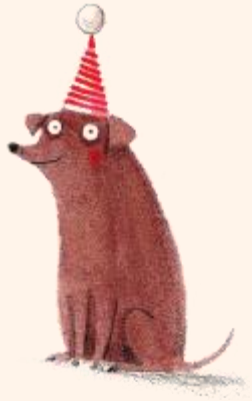
```
//例: 求语文成绩大于等于90且总成绩大于等于平均成绩的学生
//求学生总成绩的平均值
double averagescore = (from student in students
                       let totlescore = student.Scores.Sum()
                       select totlescore).Average();
Console.WriteLine($"Average score: {averagescore}");

//完成查询
IEnumerable<Student> query4 = from student in students
          where student.Scores[0]⩾90 && student.Scores.Sum() ⩾ averagescore
          select student;
```



2.3.2 orderby 子句 、join 子句

- orderby子句：可就数据中某一个或多个字段按升序或降序对结果进行排序。降序descending、升序ascending。

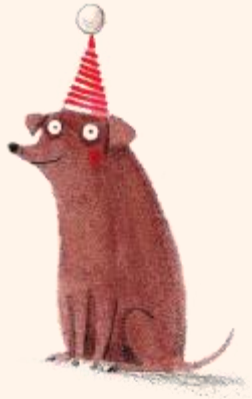
- join子句：使用 join 子句可基于每个元素中指定的键之间的相等比较，将一个数据源中的元素与另一个数据源中的元素进行关联和/或合并。



2.3.2 let 子句

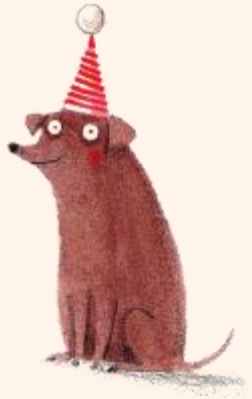
- 使用 let 子句可将表达式（如方法调用）的结果存储在新范围变量中。

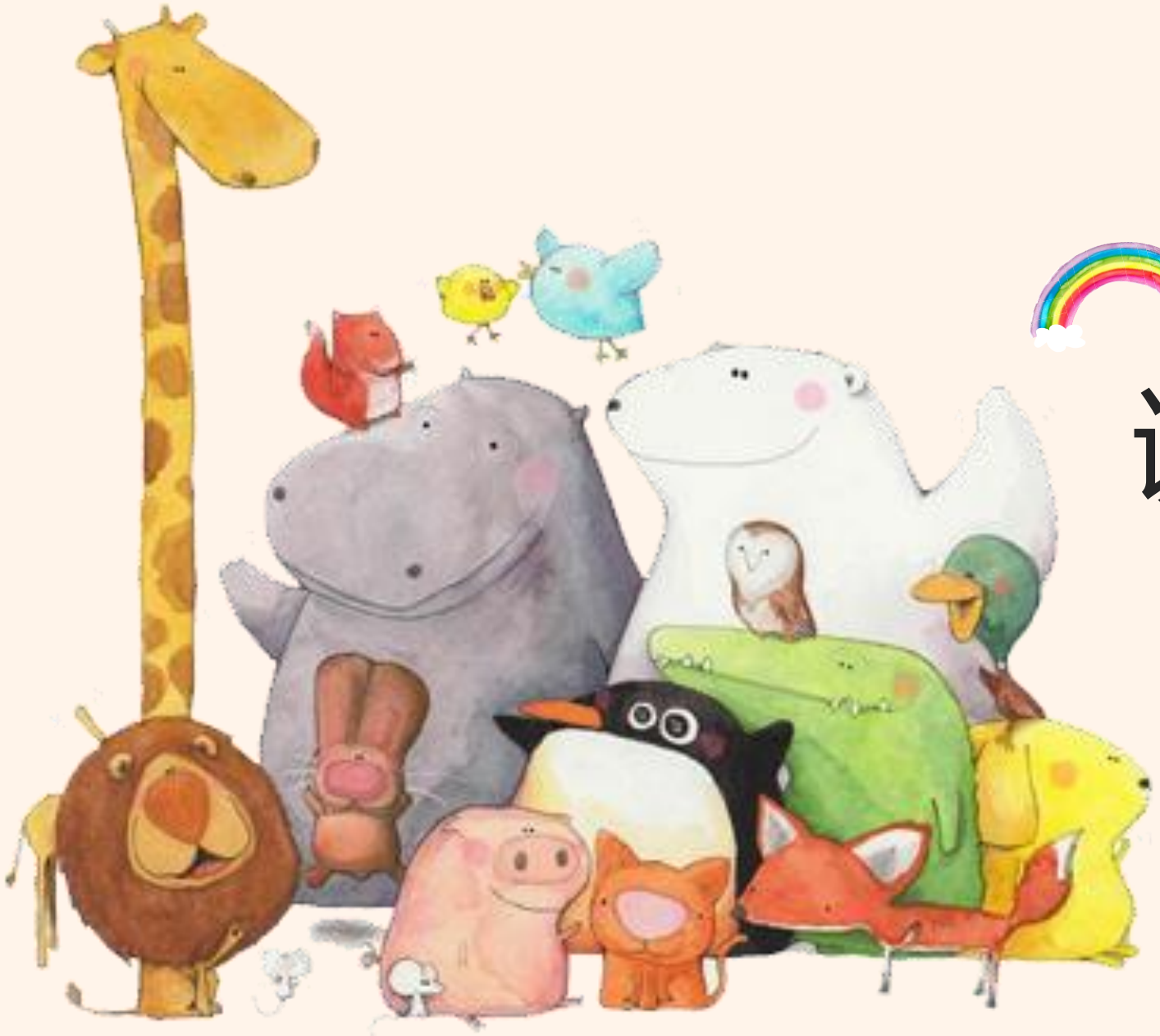
```
1 //计算每个学生的总成绩
2 IEnumerable<int> query5 = from student in students
3                           let totalscore = student.Scores.Sum()
4                           select totalscore;
5 foreach (var item in query5)
6 {
7     Console.WriteLine(item);
8 }
```



2.3.3 查询表达式中的子查询

```
// 子查询样例
// 完成：将学生根据姓氏分组，再找出每个姓氏中总成绩最高，输出姓氏及总成绩，并按总成绩降序排列
var queryGroupMax =
    from student in students
    group student by student.SurName into studentGroup
    select new
    {
        Level = studentGroup.Key,
        HighestScore = (      //子查询
            from student2 in studentGroup
            select student2.Scores.Sum()
        ).Max()
    };
Console.WriteLine("每个姓氏的最高分：");
foreach (var item in queryGroupMax)
{
  Console.WriteLine($"{item.Level}：{item.HighestScore}");
}
```





谢谢！