C# Language Integrated Query

LINQ

Orlando Campos Pasten

SQL Server DBs
XML
ADO.NET Datasets
Collections
IEnumerable

It bridges the gap between the world of objects and the world of data.

.Net Framework 3.5

Set of technologies based on the integration of query capabilities directly into the C# language.
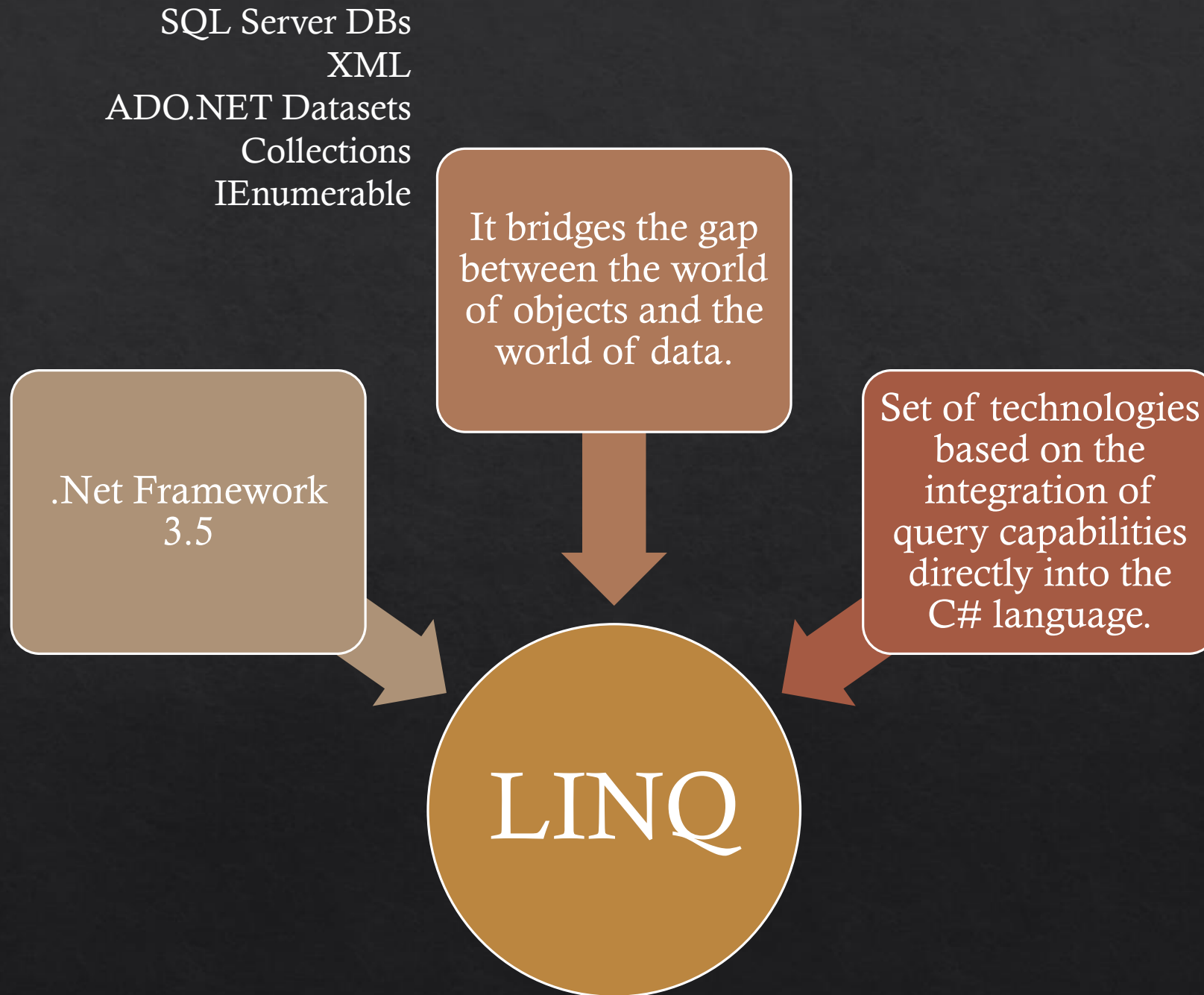
LINQ

3 Ways for writing LINQ queries

Query syntax

Method syntax

Combination (query and method syntax)

```csharp
List<int> numbers = new List<int>() { 5, 4, 1, 3, 9, 8, 6, 7, 2, 0 };

// The query variable can also be implicitly typed by using var
IEnumerable<int> filteringQuery =
    from num in numbers
    where num < 3 || num > 7
    select num;

IEnumerable<int> orderingQuery =
    from num in numbers
    where num < 3 || num > 7
    orderby num ascending
    select num;
```

```
List<int> numbers1 = new List<int>() { 5, 4, 1, 3, 9, 8, 6, 7, 2, 0 };
List<int> numbers2 = new List<int>() { 15, 14, 11, 13, 19, 18, 16, 17, 12, 10 };

double average = numbers1.Average();

IEnumerable<int> concatenationQuery = numbers1.Concat(numbers2);

// If the method has Action or Func parameters, these are provided in the form of
a lambda expression.
IEnumerable<int> largeNumbersQuery = numbers2.Where(c => c > 15);
```

Each of the previous queries can be written by using implicit typing with var
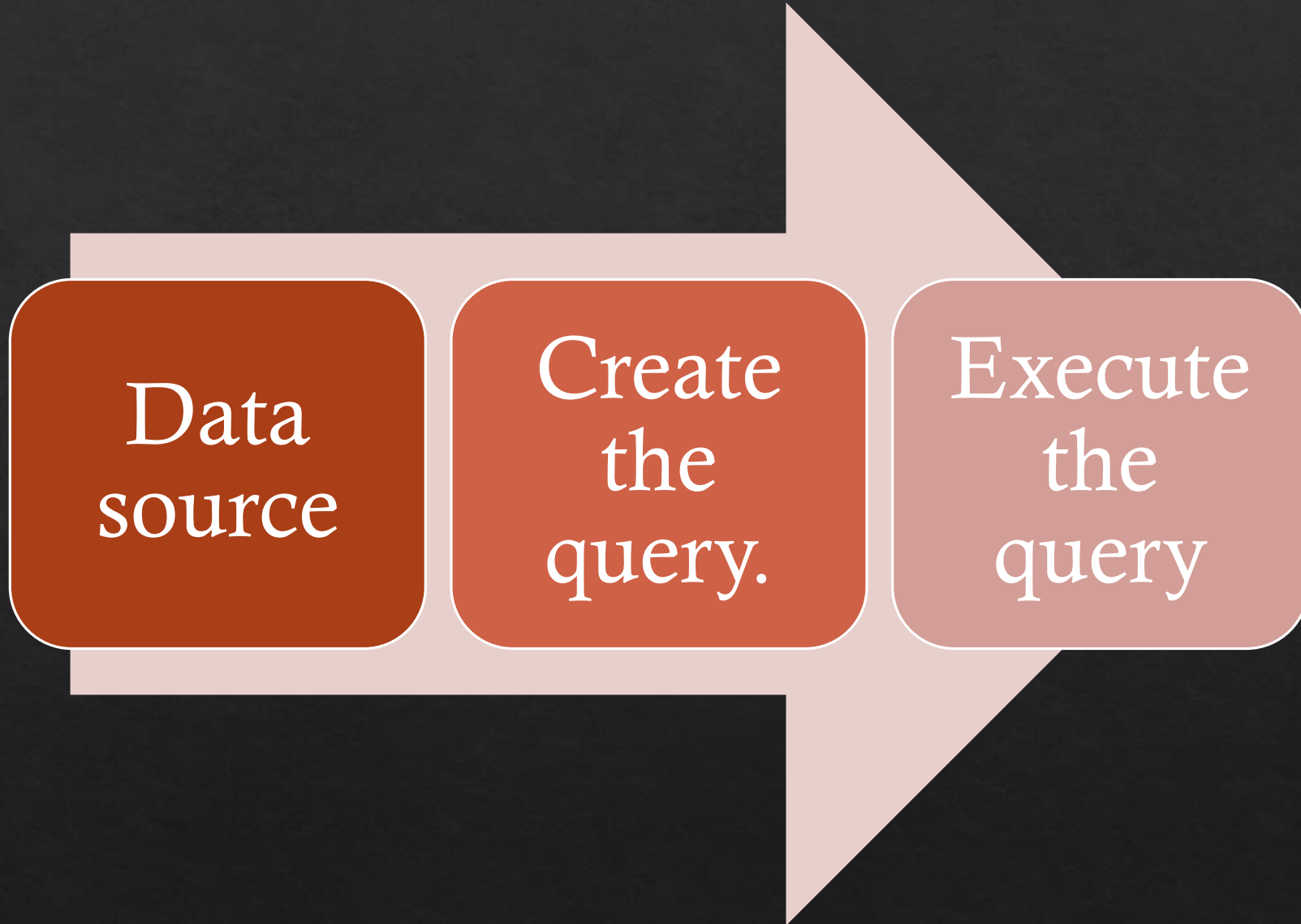
```
List<int> numbers1 = new List<int>() { 5, 4, 1, 3, 9, 8, 6, 7, 2, 0 };

public void ExecuteQuery1()
{
    // Using a query expression with method syntax
    // 1, 9, 8, 2, 0 = 5
    int numCount1 =
        (from num in numbers1
         where num < 3 || num > 7
         select num).Count();

    Console.WriteLine(numCount1);
}
```

The previous query can be written by using implicit typing with var
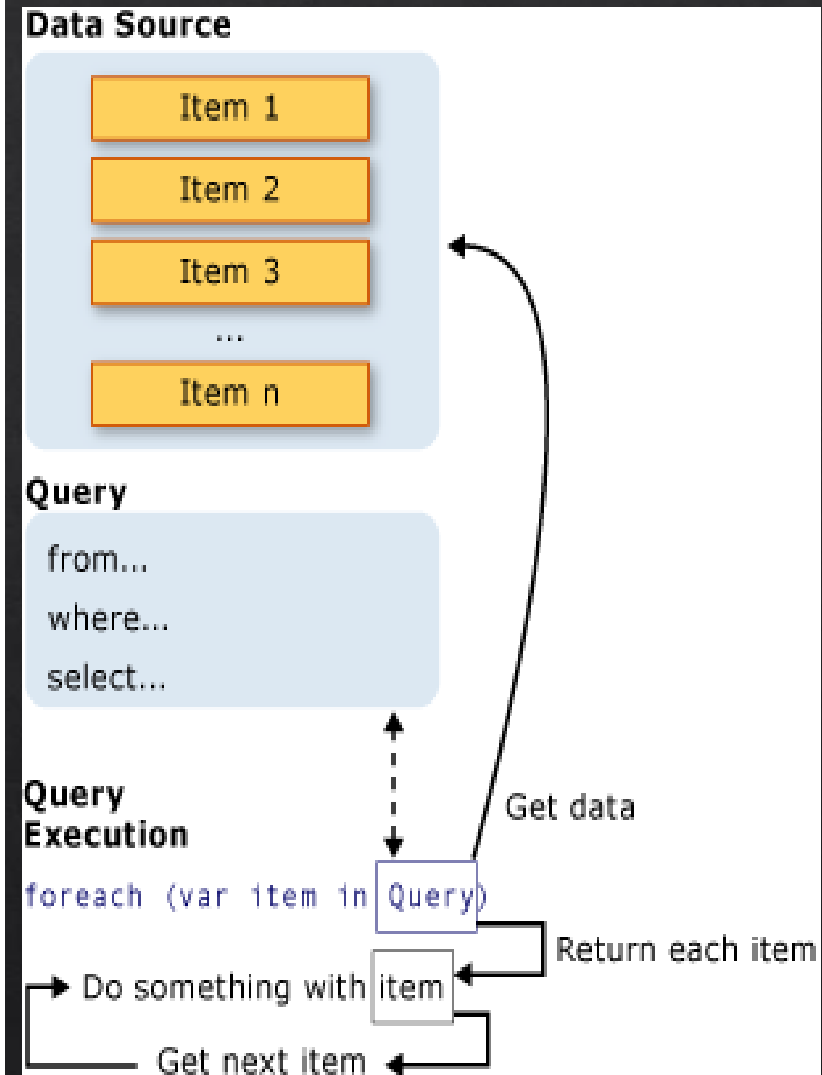
Three Parts of a Query Operation

Data source

Create the query.

Execute the query

```csharp
class IntroToLINQ
{

    static void Main()
    {

        //  1. Data source.
        int[] numbers = new int[7] { 0, 1, 2, 3, 4, 5, 6 };

        // 2. Query creation.
        // numQuery is an IEnumerable<int>
        var numQuery =
            from num in numbers
            where (num % 2) == 0
            select num;


        // 3. Query execution.
        foreach (int num in numQuery)
        {
            Console.Write("{0,1} ", num);
        }
    }
}
```


Data Source

Item 1
Item 2
Item 3
...
Item n

Query

from...
where...
select...

Query Execution

foreach (var item in Query)

Get data

Return each item

Do something with item

Get next item

Basic LINQ Query Operations