INFORMATION PROCESSING TECHNIQUES

LINQ

WEEK 14

MURTAZA MUNAWAR FAZAL

What is LINQ?

- Language Integrated Query
- Make query a part of the language
- Component of .NET Framework 3.5

Query without LINQ

Objects using loops and conditions

```
foreach(Customer c in customers)
   if (c.Region == “PAK") ...
```

Databases using SQL

```
SELECT * FROM Customers WHERE Region=‘PAK'
```

XML using XPath/XQuery

```
//Customers/Customer[@Region=‘PAK']
```

ADO without LINQ

```
SqlConnection con = new SqlConnection(...);
con.Open();
SqlCommand cmd = new SqlCommand(
            @"SELECT * FROM Customers
                WHERE c.Region = @Region", con
            );
cmd.Parameters.AddWithValue("@Region", “PAK");
DataReader dr = cmd.ExecuteReader();
while (dr.Read()) {
  string name = dr.GetString(dr.GetOrdinal("Name"));
  string phone = dr.GetString(dr.GetOrdinal("Phone"));
  DateTime date = dr.GetDateTime(3);
}
dr.Close();
con.Close();
```

LINQ to…

LINQ to Objects

LINQ to SQL (formerly known as DLINQ)

LINQ to XML (formerly known as XLINQ)

LINQ to Entities (ADO.NET Entities)

Example

- You have the following array. Return all numbers which are greater then 4.
- Int[] Values = { 2,9,5,0,3,7,1,4,8,6};

- Var filtered =   from value in Values

```
                   where value > 4
                   select value
```

Example

- Our first LINQ query begins with a From clause which specifies a range variable (`value`) and the data source to query (the array `values`).
- The range variable represents each item in the data source, much like the control variable in a `For Each`…`Next` statement.
- If the condition in the Where clause evaluates to `True`, the element is selected—that is, it’s included in the collection of `Integer`s that represents the query results.
- Here, the `Integer`s in the array are included only if they’re greater than `4`.

Example (Sorting)

- The LINQ query in the above example selects the elements of the array `values` and returns an `IEnumerable` object containing a sorted copy of the elements.

- Int[] Values = { 2,9,5,0,3,7,1,4,8,6};

- Var filtered = from value in Values
  Order By value
  select value

Query with LINQ (C#)

```
var myCustomers = from c in customers
    where c.Region == “PAK"
    select c;
```

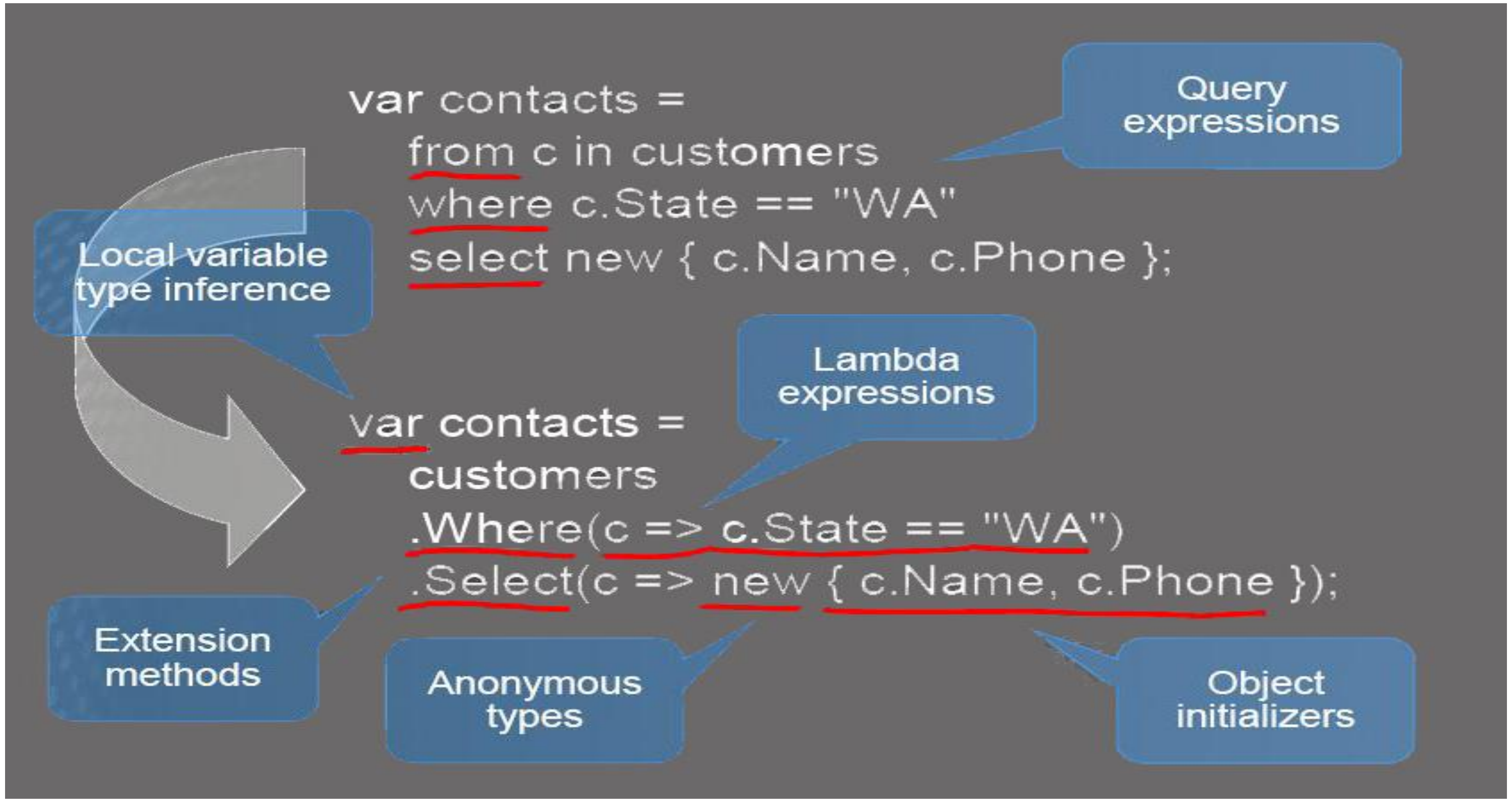
LINQ to ADO.Net

```
using (NorthwindDataContext db = new NorthwindDataContext())
{
  //You can also use “var” at “IEnumerable<Customer>”

  IEnumerable<Customer> custs = from c in db.Customers
                select c;

 foreach (Customer c in custs)
  {
   Console.WriteLine(c.CompanyName);

  }

}
```

LINQ to XML

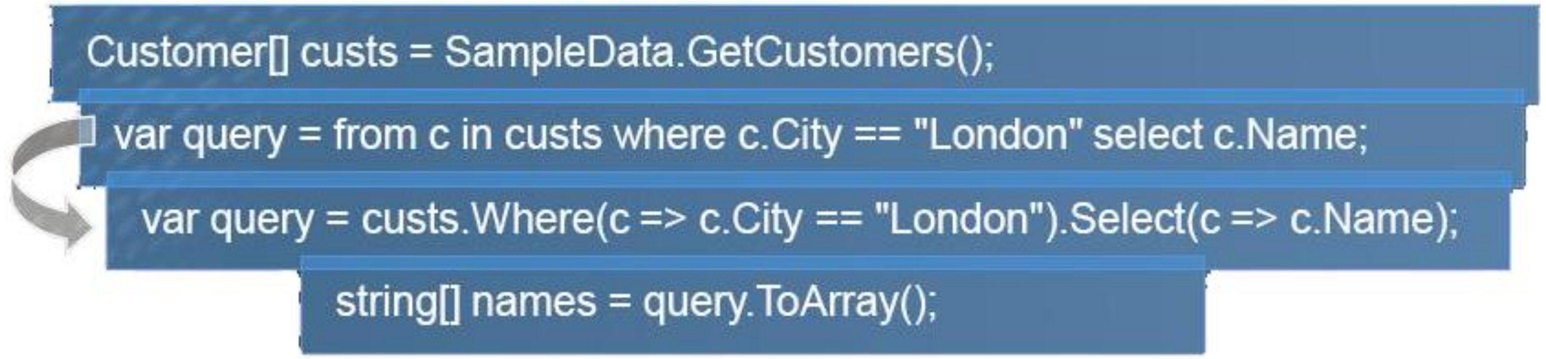
```
XElement xelement = XElement.Load("..\\..\\Employees.xml");

IEnumerable<XElement> employees = xelement.Elements();

// Read the entire XML

foreach (var employee in employees)

{

   Console.WriteLine(employee);

}
```

LINQ Prerequisite

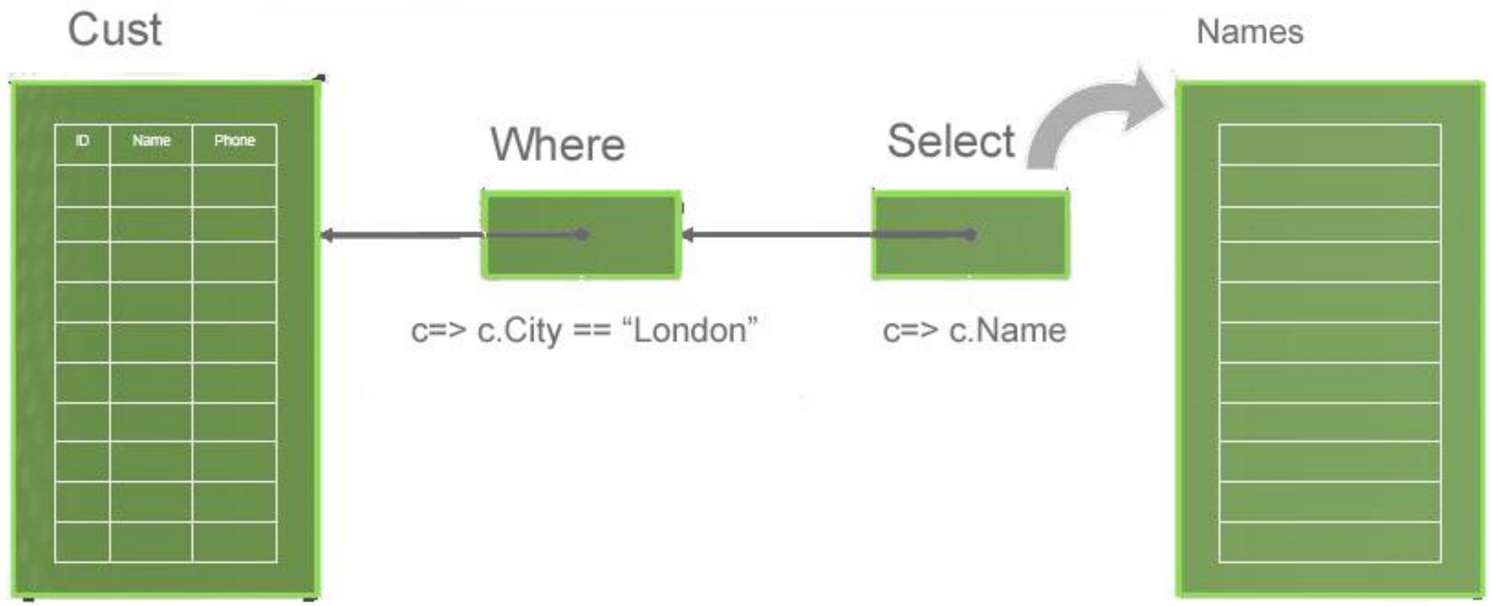


```
var contacts =
    from c in customers
    where c.State == "WA"
    select new { c.Name, c.Phone };

var contacts =
    customers
    .Where(c => c.State == "WA")
    .Select(c => new { c.Name, c.Phone });
```

Deferred Query Execution



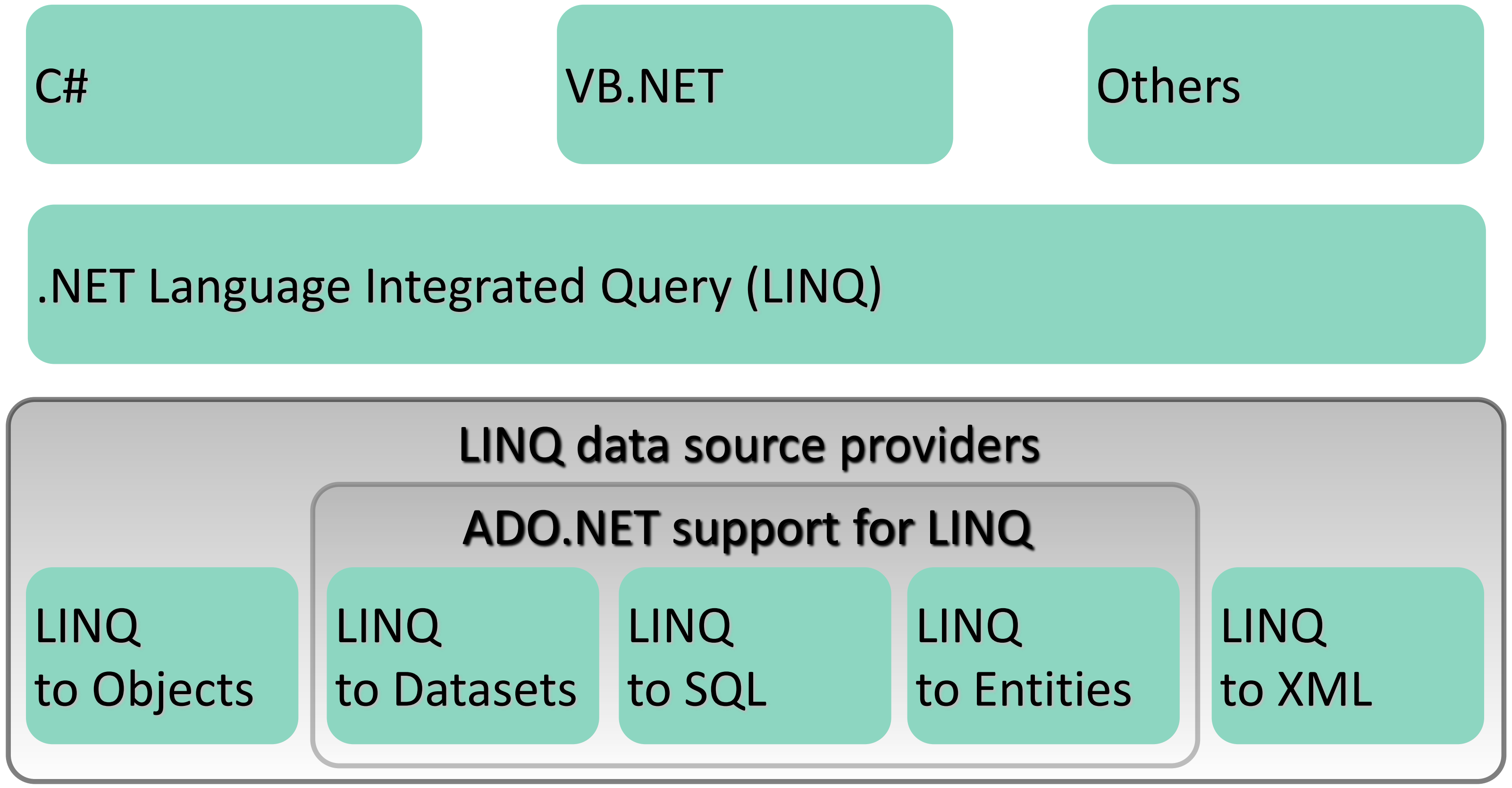
```
Customer[] custs = SampleData.GetCustomers();
var query = from c in custs where c.City == "London" select c.Name;
var query = custs.Where(c => c.City == "London").Select(c => c.Name);
string[] names = query.ToArray();
```



Advantages

- Unified data access
  Single syntax to learn and remember
- Strongly typed
  Catch errors during compilation
- IntelliSense
  Prompt for syntax and attributes
- Bindable result sets

Architecture



LINQ to Objects

```
int[] nums = new int[] {0,4,2,6,3,8,3,1};

double average = nums.Take(6).Average();
var above = from n in nums
        where n > average
        select n;
```

Querying an Array of Reference-Type Elements Using LINQ

- When you type the name of an IEnumerable object (such as an array or the result of a LINQ query) then type the dot (.) separator, the list of the methods and properties that can be used with that object are shown.
- Some of the methods are so-called extension methods.
- For example, if you have an array of Doubles called numbers and you want to calculate the average of its values, you can simply call the Average extension method, as in numbers.Average().

LINQ to Objects

- Query any IEnumerable<T> source
  Includes arrays, List<T>, Dictionary...
- Many useful operators available
  Sum, Max, Min, Distinct, Intersect, Union
- Expose your own data with IEnumerable<T> or IQueryable<T>
- Create operators using extension methods

LINQ operators

Aggregate	Conversion	Ordering	Partitioning	Sets
Aggregate	Cast	OrderBy	Skip	Concat
Average	OfType	ThenBy	SkipWhile	Distinct
Count	ToArray	Descending	Take	Except
Max	ToDictionary	Reverse	TakeWhile	Intersect
Min	ToList			Union
Sum	ToLookup			
	ToSequence			

and many others

LINQ to SQL (formerly known as Dlinq)

- Object-relational mapping
  Records become strongly-typed objects
- Data context is the controller mechanism
- Facilitates update, delete & insert
- Translates LINQ queries behind the scenes
- Type, parameter and injection safe

Limitations

LINQ

Only defines query, not update or context

LINQ To SQL

- limited to SQL Server as backend
- requires at least .NET 3.5 to run
- somewhat limited in that tables are mapped strictly on a 1:1 basis (one table = one class)

.NET features used

.NET Framework 2.0

- Partial classes (mapping)

.NET Framework 3.5

- Anonymous types (shaping)
- Extension methods (query operators)
- Type inference (var keyword)
- Lambda expressions (query syntax)

More LINQ queries

```
var goodCusts = (from c in db.Customers
    where c.PostCode.StartsWith(“PK")
    orderby c.Sales descending
    select c).Skip(10).Take(10);
```