

Evoluzione del linguaggio C# (Visual Studio 2011)

```

public async Task<XElement> GetXmlAsync(string url) {
    var client = new HttpClient();
    var response = await client.GetAsync(url);
    var text = response.Content.ReadAsStringAsync();
    return XElement.Parse(text);
}

public Task<XElement> GetXmlAsync(string url) {
    var tcs = new TaskCompletionSource<XElement>();
    var client = new HttpClient();
    client.GetAsync(url).ContinueWith(task => {
        var response = task.Result;
        var text = response.Content.ReadAsStringAsync();
        tcs.SetResult(XElement.Parse(text));
    });
    return tcs.Task;
}

```

C# 5.0

Windows runtime
Asynchronous
programming

C# 4.0

Dynamic type
Interoperability

- **Metodi asincroni**
trasformano il codice
automaticamente in una
callback state machine

Il linguaggio Linq

3

Il progetto Microsoft LINQ

- ✦ **Problema: Data != Objects**
- ✦ **LINQ (Language INtegrated Query)** è un set di estensioni per .NET Framework che comprende operazioni di query, d'impostazione e di trasformazione. Estende C# e Visual Basic con sintassi del linguaggio nativa per le query e offre librerie di classi che consentono di sfruttare al meglio tali funzionalità.
- ✦ **Benefici in C# e VB**
 - ✦ unica modalità d'interrogazione di oggetti, tabelle relazionali, documenti XML;
 - ✦ Type checking and IntelliSense per le query;
 - ✦ espressività SQL e Xquery-like in C# e VB;
 - ✦ modello esteso per linguaggi/API.
- ✦ **LINQ to XML** è stato sviluppato tenendo principalmente in considerazione l'applicazione a XML e sfrutta operatori di query standard oltre ad aggiungere estensioni di query specifiche per XML.

Riferimento : <http://msdn.microsoft.com/netframework/future/linq/>

Il linguaggio Linq

4

Linq: concetti di base

- **Language Enhancements**

Local Variable Type Inference

Object Initializers

Anonymous Types

Lambda Expressions

Extension Methods

+ **Query Expressions**

= LINQ ☺

Il linguaggio Linq



5

Local Variable Type Inference

```
int i = 2010;  
string s = "Ciao";  
double d = 3.14;  
int[] numbers = new int[] {10, 20, 30};  
Dictionary<int,Order> orders = new Dictionary<int,Order>();
```

```
var i = 2010;  
var s = "Ciao";  
var d = 3.14;  
var numbers = new int[] {10, 20, 30};  
var orders = new Dictionary<int,Order>();
```

"Il tipo è dedotto dalla parte a destra della dichiarazione"

Il linguaggio Linq



6

Object Initializers

```

public class Point
{
    private int x, y;

    public int X { get { return x; } set { x = value; } }
    public int Y { get { return y; } set { y = value; } }
}

```

Field or property assignments

```

Point myPoint = new Point { X = 0, Y = 1 };

```

Autoimplemented properties

```


public class Point
{
    public int X { get; set; }
    public int Y { get; set; }
}

```

```

Point myPoint = new Point();
myPoint.X = 0;
myPoint.Y = 1;

```

Il linguaggio Linq 

7

Anonymous Types

```

class X
{
    public string Name;
    public int Age;
}


```

X

```

var p = new { Name = "Betty", Age = 31 };

```

Il linguaggio Linq 

8

Lambda Expressions (1)

✚ Un'espressione lambda è una funzione anonima che può contenere espressioni e istruzioni e che può essere utilizzata per creare delegati o tipi di struttura ad albero dell'espressione. Tutte le espressioni lambda utilizzano l'operatore lambda \Rightarrow , che è letto come "goes to".

✚ Il lato sinistro dell'operatore lambda specifica i parametri di input, se presenti, e il lato destro contiene l'espressione o il blocco di istruzioni.

✚ L'espressione lambda $x \Rightarrow x * x$ viene letta "x goes to x times x". Questa espressione può essere assegnata a un tipo delegato:

```
delegate int del(int i);
static void Main(string[] args)
{
    del myDelegate = x => x * x;
    int j = myDelegate(6);    // j = 36
}
```

Il linguaggio Linq



9

Lambda Expressions (2)

✚ Un altro esempio:

```
List<int> numbers = new List<int> { 1, 2, 3, 4, 5, 6, 7 };
```

```
Func<int, bool> where = n => n < 6;
```

Lambda
expression

```
Func<int, string> orderby = n => n % 2 == 0 ? "even" : "odd";
```

Lambda
expression

```
var nums = numbers.Where(where).OrderBy(orderby);
```

Risultato
nums = 2,4,1,3,5

Il linguaggio Linq




10

Lambda Expr. vs delegate (1)

```
public delegate bool Predicate<T>(T obj);

public class List<T>
{
    public List<T> FindAll(Predicate<T> test) {
        List<T> result = new List<T>();
        foreach (T item in this)
            if (test(item)) result.Add(item);
        return result;
    }
    ...
}
```

Modo
convenzionale
con delegate

Il linguaggio Linq 


11

Lambda Expr. vs delegate (2)

```
public class MyClass
{
    public static void Main() {
        List<Customer> customers = GetCustomerList();
        List<Customer> locals =
            customers.FindAll(
                new Predicate<Customer>(CityEqualsLondon)
            );
    }

    static bool CityEqualsLondon(Customer c) {
        return c.City == "LONDON";
    }
}
```

Modo
convenzionale
con delegate


Il linguaggio Linq 

12

Lambda Expr. vs delegate (3)

```
public class MyClass
{
    public static void Main() {
        List<Customer> customers = GetCustomerList();
        List<Customer> locals =
            customers.FindAll(
                delegate(Customer c) { return c.City == "LONDON"; }
            );
    }
}
```

Modo
convenzionale
con delegate


Il linguaggio Linq 

13

Lambda Expr. vs delegate (4)

```
public class MyClass
{
    public static void Main() {
        List<Customer> customers = GetCustomerList();
        List<Customer> locals =
            customers.FindAll(
                (Customer c) => {return c.City == "LONDON";}
            );
    }
}
```

Lambda
expression


Il linguaggio Linq 

14

Lambda Expr. vs delegate (5)

```
public class MyClass
{
    public static void Main() {
        List<Customer> customers = GetCustomerList();
        List<Customer> locals =
            customers.FindAll(
                c => c.City == "LONDON"
            );
    }
}
```


Lambda expression

Il linguaggio Linq  15

Lambda Expr. vs delegate (6)

```
public class MyClass
{
    public static void Main() {
        List<Customer> customers = GetCustomerList();
        List<Customer> locals =
            customers.FindAll(c => c.City == "LONDON");
    }
}
```

Lambda expression

Il linguaggio Linq  16

Lambda Expressions: altri esempi

```

int[] v = new int[10] { 3, 1, 12, 54, 8, 34, 5, 19, 25, 30};
int[] idx1 = Array.FindAll<int>(v, item => item > 8);
int[] idx2 = Array.FindAll<int>(v, item => item < 20 && item > 5);

string[] digits = { "zero", "one", "two", "three", "four", "five", "six",
                    "seven", "eight", "nine" };
var shortDigits = digits.Where((digit, index) => digit.Length < index);

string[] words = { "cherry", "apple", "blueberry" };
int shortestWordLength = words.Min(w => w.Length);


```

idx1 = 12, 54, 34, 19, 25, 30

idx2 = 12, 8, 19

shortDigits = five, six, seven, eight, nine

shortestWordLength = 5

Il linguaggio Linq  17

Extension Methods

```

namespace StringExtensions
{
    public static class StringExtensionsClass
    {
        public static string RemoveNonNumeric(this string s)
        {
            MatchCollection col = Regex.Matches(s, "[0-9]");
            StringBuilder sb = new StringBuilder();
            foreach (Match m in col) sb.Append(m.Value);
            return sb.ToString();
        }
    }
}

```


Si estende la classe String con il metodo RemoveNonNumeric

```

using StringExtensions;
....
....
string phone = "123-123-1234";
string newPhone = phone.RemoveNonNumeric();

```

Esempio di uso

Il linguaggio Linq  18

Query Expressions --> Linq

- Query che invocano metodi
 - Where, Join, OrderBy, Select, GroupBy, ...

```

from c in customers
where c.City == "LONDON"
select new { c.Name, c.Phone };

```

```

customers
.Where(c => c.City == "LONDON")
.Select(c => new { c.Name, c.Phone });

```

Il linguaggio Linq

19

Linq in sintesi

C# 3.0

VB 9.0

Others...

.NET Language-Integrated Query

LINQ enabled data sources

LINQ
To Objects

LINQ enabled ADO.NET

LINQ
To DataSets

LINQ
To SQL

LINQ
To Entities

LINQ
To XML

Objects

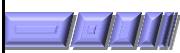
SQL WinFS

<book>
 <title/>
 <author/>
 <year/>
 <price/>
 </book>

XML

Il linguaggio Linq

20



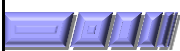
ORM: Object Relational Mapping

ORM è una tecnica che favorisce l'integrazione di sistemi software aderenti al **paradigma di programmazione a oggetti** con sistemi **RDBMS**. Un prodotto ORM fornisce, mediante un'interfaccia orientata agli oggetti i servizi per la **persistenza dei dati**, astruendo al contempo dalle caratteristiche implementative dello specifico RDBMS utilizzato.

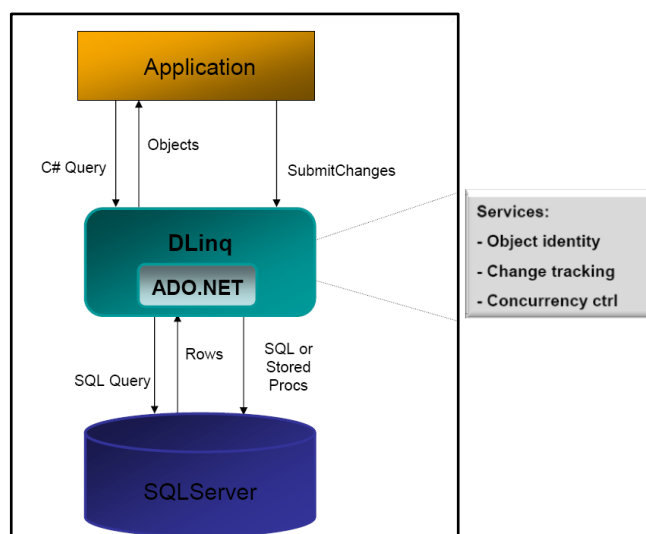
Vantaggi:

- + superamento (più o meno completo) dell'**incompatibilità tra progettazione orientata agli oggetti e modello relazionale per la rappresentazione dei dati**;
- + elevata portabilità rispetto alla tecnologia DBMS utilizzata;
- + sensibile riduzione dei tempi di sviluppo del codice;
- + approccio stratificato, isolando in un solo livello la logica di persistenza dei dati, a vantaggio della modularità complessiva del sistema.

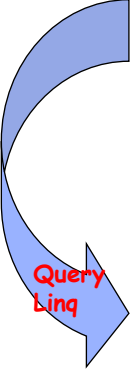
Microsoft offre un ORM di base in C# 3.0 con Linq e Visual Studio Designer.



ORM: Object Relational Mapping



Stile di programmazione LINQ



```


class Contact { ... }
List<Contact> contacts = new List<Contacts>();
foreach(Customer c in customers)
{
    if(c.State == "CA")
    {
        Contact ct = new Contact();
        ct.Name = c.Name;
        ct.Email = c.Email;
        contacts.Add(ct);
    }
}

```

```

var contacts =
    from c in customers
    where c.State == "CA"
    select new { c.Name, c.Email};

```

Il linguaggio Linq 
23

Sintassi di una query Linq

```

from id in source
{ from id in source |
join id in source on expr equals expr [ into id ] |
let id = expr |
where condition |
orderby ordering, ordering, ... }
select expr | group expr by key
[ into id query ]


```


Starts with
from

Zero or more
from, join, let,
where, or
orderby

Ends with
select or group
by

Optional *into*
continuation

Il linguaggio Linq 
24




Vari tipi di operatori


[Sorting Data](#)
[Set Operations](#)
[Filtering Data](#)
[Quantifier Operations](#)
[Projection Operations](#)
[Partitioning Data](#)
[Join Operations](#)
[Grouping Data](#)
[Generation Operations](#)
[Equality Operations](#)
[Element Operations](#)
[Converting Data Types](#)
[Concatenation Operations](#)
[Aggregation Operations](#)

<http://msdn.microsoft.com/en-us/library/bb397896.aspx>

Il linguaggio Linq




25



Esempi di operatori

Restriction	Where
Projection	Select, SelectMany
Ordering	OrderBy, ThenBy
Grouping	GroupBy
Quantifiers	Any, All
Partitioning	Take, Skip, TakeWhile, SkipWhile
Sets	Distinct, Union, Intersect, Except
Elements	First, FirstOrDefault, ElementAt
Aggregation	Count, Sum, Min, Max, Average
Conversion	ToArray, ToList, ToDictionary
Casting	OfType<T>
Join	<expr> Join <expr> On <expr> Equals <expr>

Il linguaggio Linq



26

ADO.Net per query su relazioni

```


SqlConnection c = new SqlConnection(...);
c.Open();
SqlCommand cmd = new SqlCommand(
    @"SELECT c.Name, c.Email
    FROM Customers c
    WHERE c.City = @p0");
cmd.Parameters.AddWithValue("@p0", "London");
DataReader dr = c.Execute(cmd);
while (dr.Read())
{
    string name = dr.GetString(0);
    string email = dr.GetString(1);
    // show results
    .....
}
dr.Close();

```

Queries in quotes

Loosely bound arguments

Loosely typed result sets

Il linguaggio Linq  27

DLinq per query su tabelle

```

DataClassesNorthwindDataContext db = new DataClassesNorthwindDataContext();

var myQuery =
    from p in db.Products
    where p.UnitsInStock < p.ReorderLevel && !p.Discontinued
    select p;


```

Strongly typed connection

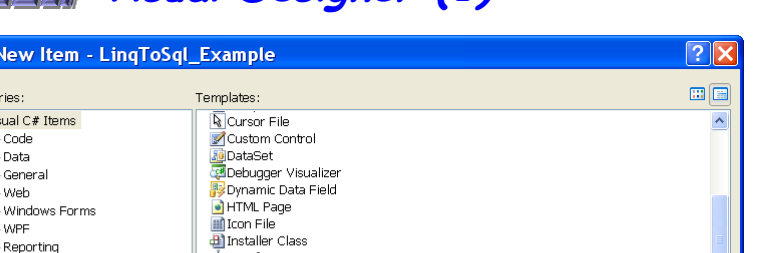
Tables are like collections

Integrated query syntax

Strongly typed results

Il linguaggio Linq  28

Visual Designer (1)



Visual Designer (1)

Add New Item - LingToSql_Example

Categories:

- Visual C# Items
 - Code
 - Data
 - General
 - Web
 - Windows Forms
 - WPF
 - Reporting
 - Workflow
 - XNA Game Studio 3.0
 - XNA Game Studio 3.1

Templates:

- Cursor File
- Custom Control
- DataSet
- Debugger Visualizer
- Dynamic Data Field
- HTML Page
- Icon File
- Installer Class
- Interface
- JScript File
- LINQ to SQL Classes**
- Local Database
- Local Database Cache
- MDI Parent Form
- Report
- Report Wizard
- Resources File
- Service-based Database

LINQ to SQL classes mapped to relational objects.

Name: NorthWindDataClasses.dbml

Add Cancel

Il linguaggio Ling

29



Visual Designer (3)

Classe generata automaticamente che consente di interfacciarsi al DB

```

namespace LinqToSql_Example
{
    using System.Data.Linq;
    using System.Data.Linq.Mapping;
    using System.Data;
    using System.Collections.Generic;
    using System.Reflection;
    using System.Linq;
    using System.Linq.Expressions;
    using System.ComponentModel;
    using System;

    [System.Data.Linq.Mapping.DatabaseAttribute(Name="NORTHWND")]
    public partial class NorthWindDataClassesDataContext : System.Data.Linq.DataContext
    {
        private static System.Data.Linq.Mapping.MappingSource mappingSource
            = new AttributeMappingSource();

        #region Extensibility Method Definitions
        partial void OnCreated();
        partial void InsertCategory(Category instance);
        partial void UpdateCategory(Category instance);
        partial void DeleteCategory(Category instance);
        partial void InsertProduct(Product instance);
    }
}

```

Il linguaggio Linq

31

Le classi per il mapping O-R

```

[Table(Name="dbo.Products")]
public partial class Product : INotifyPropertyChanging, INotifyPropertyChanged
{
    private static PropertyChangingEventArgs emptyChangingEventArgs =
        new PropertyChangingEventArgs(String.Empty);
    private int _ProductID;
    private string _ProductName;
    private System.Nullable<int> _SupplierID;
    private System.Nullable<int> _CategoryID;
    private string _QuantityPerUnit;
    private System.Nullable<decimal> _UnitPrice;
    private System.Nullable<short> _UnitsInStock;
    private System.Nullable<short> _UnitsOnOrder;
    private System.Nullable<short> _ReorderLevel;
    private bool _Discontinued;
    private EntitySet<Order_Detail> _Order_Details;
    private EntityRef<Category> _Category;

    .....

[Table(Name="dbo.Categories")]
public partial class Category : INotifyPropertyChanging, INotifyPropertyChanged
{
    private static PropertyChangingEventArgs emptyChangingEventArgs =
        new PropertyChangingEventArgs(String.Empty);
    private int _CategoryID;
    private string _CategoryName;
    private string _Description;
    private System.Data.Linq.Binary _Picture;
    private EntitySet<Product> _Products;

    #region Extensibility Method Definitions
    partial void OnLoaded();
    .....
}

```

Il linguaggio Linq

32

Esempi di query Linq (1)

Si suppone:

- di aver inserito nella form di presentazione dei risultati un controllo denominato **dataGridViewQuery** della classe **System.Windows.Forms.DataGridView**;
- di aver dichiarato **NorthWindDataClassesDataContext db**;
- di aver istanziato il contesto **db = new NorthWindDataClassesDataContext();**

Implicitly Typed Local Variable

```
var productsQuery = from p in db.Products
                    where p.UnitPrice > 50
                    select p;
```

Elenco dei prodotti il cui prezzo unitario è maggiore di 50 \$

```
// binding della query con il dataGridViewQuery della form
dataGridViewQuery.DataSource = productsQuery;
// si cancella l'ultima colonna che rappresenta il link a category
dataGridViewQuery.Columns.Remove(dataGridViewQuery.Columns[dataGridViewQuery.Columns.Count - 1]);
```

	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitInStk
▶	Mishi Kobe Niku	4	6	18 - 500 g pkgs.	97.0000	29
	Carnarvon Tigers	7	8	16 kg pkg.	62.5000	42
	Sir Rodney's Mar...	8	3	30 gift boxes	81.0000	40
	Thüringer Rostbr...	12	6	50 bags x 30 sau...	123.7900	0
	Côte de Blaye	18	1	12 - 75 cl bottles	263.5000	17
	Manjimup Dried A...	24	7	50 - 300 g pkgs.	53.0000	20
	Raclette Courdav...	28	4	5 kg pkg.	55.0000	79
*						

dataGridViewQuery

Il linguaggio Linq

33

Esempi di query Linq (2)

```
var productsQuery = from p in db.Products
                    where String.Compare(p.Category.CategoryName, "M", true) > 0
                    select p;
```

Elenco dei prodotti per cui il nome della categoria è maggiore della lettera "M"

```
// binding della query con il dataGridViewQuery della form
dataGridViewQuery.DataSource = productsQuery;
// si cancella l'ultima colonna che rappresenta il link a category
dataGridViewQuery.Columns.Remove(dataGridViewQuery.Columns[dataGridViewQuery.Columns.Count - 1]);
```

	ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice
▶	9	Mishi Kobe Niku	4	6	18 - 500 g pkgs.	97.0000
	17	Alice Mutton	7	6	20 - 1 kg tins	39.0000
	29	Thüringer Rostbr...	12	6	50 bags x 30 sau...	123.7900
	53	Perth Pasties	24	6	48 pieces	32.8000
	54	Tourtière	25	6	16 pies	7.4500
	55	Pâté chinois	25	6	24 boxes x 2 pies	24.0000
	7	Uncle Bob's Orga...	3	7	12 - 1 lb pkgs.	30.0000

Il linguaggio Linq

34

```
var productsQuery = (from p in db.Products
                    where p.Category.CategoryName.StartsWith("D")
                    select p).Take(5);
dataGridViewQuery.DataSource = productsQuery;
dataGridViewQuery.Columns.Remove(dataGridViewQuery.Columns[dataGridViewQuery.Columns.Count - 1]);
```

I primi 5 prodotti per cui il nome della categoria inizia con la lettera "D"

	ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice
▶	11	Queso Cabrales	5	4	1 kg pkg.	21,0000
	12	Queso Manchego	5	4	10 - 500 g pkgs.	38,0000
	31	Gorgonzola Telino	14	4	12 - 100 g pkgs	12,5000
	32	Mascarpone Fabioli	14	4	24 - 200 g pkgs.	32,0000
	33	Geiest	15	4	500 g	2,5000
✱						



SQL
generato ed
eseguito
tramite
ADO.NET

```
(SELECT TOP 5 [t0].[ProductID], [t0].[ProductName], [t0].[SupplierID], [t0].[CategoryID], [t0].[QuantityPerUnit], [t0].[UnitPrice], [t0].[UnitsInStock], [t0].[UnitsOnOrder], [t0].[ReorderLevel], [t0].[Discontinued]
FROM [dbo].[Products] AS [t0]
LEFT OUTER JOIN [dbo].[Categories] AS [t1] ON [t1].[CategoryID] = [t0].[CategoryID]
WHERE [t1].[CategoryName] LIKE @p0
}
```



35

```
var resultProductsQuery =
    from p in db.Products
    where p.Category.Products.Count > 1
    select new
    (
        PID = p.ProductID,
        PN = p.ProductName,
        TotalUnits = p.Order_Details.Sum(o => o.Quantity),
        Revenue = p.Order_Details.Sum(o => o.UnitPrice*o.Quantity)
    );

dataGridViewMyType.DataSource = resultProductsQuery;
```

PID	PN	TotalUnits	Revenue
1	Chai	828	14277.600000
2	Chang	1057	18559.200000
24	Guaraná Fantástico	1125	4782.600000
34	Sasquatch Ale	506	6678.000000
35	Steeleye Stout	883	14536.800000
38	Côte de Blaye	623	149384.200000
39	Chartreuse verte	793	13150.800000
43	Iphoh Coffee	580	25079.200000
70	Outback Lager	817	11472.000000
67	Laughing Lumber...	184	2562.000000
75	Rhinôbrai Klöster...	1155	8650.550000
76	Lakkaalkkooni	981	16734.000000
77	Original Frankfurt...	791	9685.000000
61	Sirop d'érable	603	16438.800000

Per le categorie che contengono più di un prodotto visualizzare per ciascun prodotto la quantità totale ordinata e il relativo ricavo.



36

```

    erDiagram
        CUSTOMER ||--}| ORDER : "places"
        CUSTOMER {
            string CustomerID PK
            string CompanyName
            string ContactName
            string ContactTitle
            string Address
            string City
            string Region
            string PostalCode
            string Country
            string Phone
            string Fax
        }
        ORDER {
            string OrderID PK
            string CustomerID FK
            string EmployeeID
            string OrderDate
            string RequiredDate
            string ShippedDate
            string ShipVia
            float Freight
            string ShipName
            string ShipAddress
            string ShipCity
            string ShipRegion
            string ShipPostalCode
            string ShipCountry
        }
  
```

	CustomerID	Name	OrderCount	SumFreight
▶	ERNSH	Roland Mendel	30	6205,3900
	QUICK	Horst Kloss	28	5605,6300
	SAVEA	Jose Pavarotti	31	6683,7000

	ContactName	OrderDate
►	Karl Jablonski	01/05/1998
	Karl Jablonski	17/04/1998
	Karl Jablonski	24/02/1998
	Karl Jablonski	30/01/1998
	Helvetius Nagy	08/01/1998
	Karl Jablonski	13/11/1997
	Karl Jablonski	30/10/1997
	Karl Jablonski	08/10/1997
	Karl Jablonski	06/10/1997
	Karl Jablonski	11/07/1997
	Helvetius Nagy	23/06/1997
	Helvetius Nagy	19/06/1997

Il linguaggio Ling

	CustomerName	Country	City	TotalRevenue
▶	Alyndria Camino	Spain	Madrid	1467,290000
	Alexander Feuser	Germany	Leipzig	5042,200000
	Ana Trujillo	Mexico	Mexico D.F.	1042,950000
	Anabela Doming	Brazil	Sao Paulo	7310,620000
	André Fonseca	Brazil	Campanas	8702,230000
	Ann Devion	UK	London	15033,660000
	Annette Roulet	France	Toulouse	10272,350000
	Antonio Moreno	Mexico	Mexico D.F.	7515,350000
	Aria Cruz	Brazil	Sao Paulo	4438,900000
	Art Braunschweiger	USA	Lander	12489,700000
	Bernardo Batista	Brazil	Rio de Janeiro	6573,630000
	Carmine Schmitt	France	Nantes	3172,160000
	Carlos González	Venezuela	Barguimieto	17825,060000
	Carlos Hernández	Venezuela	San Ciriobal	23611,580000
	Catherine Devey	Belgium	Bruuxelles	10430,580000
	Christina Berglund	Sweden	Luleå	26368,150000
	Daniel Tonini	France	Versailles	1392,050000



Aggiorna il prezzo unitario di tutti i prodotti di categoria 4, aumentandolo del 10%.

```
// Ask the DataContext to save all the changes.
db.SubmitChanges();
```



Linq : ereditarietà

Vehicle
Class
Fields: Key, MfgPlant, VIN

Car
Class
↳ Vehicle
Fields: ModelName, TrimCode

Truck
Class
↳ Vehicle
Fields: Axles, Tonnage

Mapping di
ereditarietà
in Linq

```

[Table]
[InheritanceMapping(Code = "C", Type = typeof(Car))]
[InheritanceMapping(Code = "T", Type = typeof(Truck))]
[InheritanceMapping(Code = "V", Type = typeof(Vehicle),
    IsDefault = true)]
public class Vehicle
{
    [Column(IsDiscriminator = true)]
    public string DiscKey;
    [Column(IsPrimaryKey = true)]
    public string VIN;
    [Column]
    public string MfgPlant;
}
public class Car : Vehicle
{
    [Column]
    public int TrimCode;
    [Column]
    public string ModelName;
}
public class Truck : Vehicle
{
    [Column]
    public int Tonnage;
    [Column]
    public int Axles;
}

```

```

var q = db.Vehicle.Where(p => p is Truck);

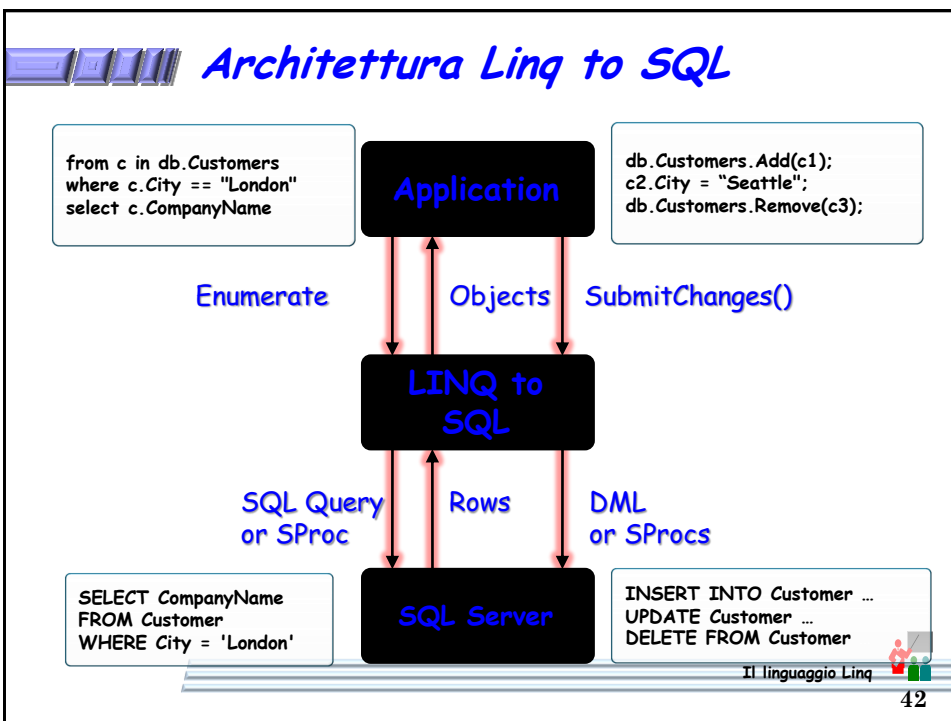
var q = db.Vehicle.OfType<Truck>();

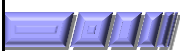
var q = db.Vehicle.Select(p => p as Truck).Where(p => p != null);
foreach (Truck p in q)
    Console.WriteLine(p.Axles);

```

Il linguaggio Linq

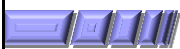
41





La classe DataContext

- ✦ A un primo sguardo sembra simile alla connessione tramite ADO.NET:
 - ✦ interrogazioni;
 - ✦ comandi;
 - ✦ transazioni;
- ✦ ma in realtà svolge molti ulteriori compiti:
 - ✦ provvede all'accesso a oggetti in un contesto object-relational;
 - ✦ effettua generazione automatica di codice
 - ✦ offre funzionalità di caching e tracking
 - ✦ ...



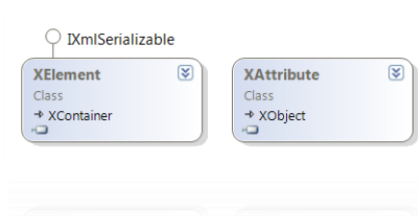
LINQ to XML

- ✦ **Supporto per:**
 - ✦ creare documenti XML;
 - ✦ leggere, interrogare, modificare e salvare documenti XML;
 - ✦ streaming, schema, annotazioni, eventi.
- ✦ Nuove XML API nella versione .NET 3.5
 - ✦ System.Xml.Linq.dll
- ✦ Namespaces
 - ✦ System.Xml.Linq
 - ✦ System.Xml.Schema
 - ✦ System.Xml.XPath
- ✦ API che possono essere usate indipendentemente da LINQ



Classi principali in System.Xml.Linq

- System.Xml.Linq è un'interfaccia API "DOM like":
 - manipola un albero XML in memoria
- Naturalmente può essere impiegata sia con interi documenti XML sia con frammenti XML.
- Le due classi chiave in System.Xml.Linq:



The diagram shows two classes: XElement and XAttribute. XElement is a class that inherits from IXmlSerializable and has a reference to XContainer. XAttribute is a class that inherits from XElement and has a reference to XObject.

Il linguaggio Linq

45

Leggere un contenuto Xml

- Per leggere in memoria XML:
 - XElement.Load
 - XDocument.Load
- Loading da:
 - URI, XmlReader, TextReader

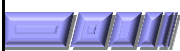
```

XmlReader reader = XmlReader.Create("myDoc.xml");
XElement element = XElement.Load(reader);

```

Il linguaggio Linq

46

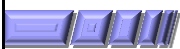


Modificare un documento XML

- ✦ Un XML tree esposto da **XElement** è modificabile.
- ✦ Le modifiche si effettuano facendo ricorso a metodi quali:
 - ✦ **XElement.Add()**
 - ✦ **XElement.Remove()**
 - ✦ **XElement.ReplaceWith()**
- ✦ Un XML tree può essere reso persistente con i metodi:
 - ✦ **XElement.Save()**, **XDocument.Save()**
 - ✦ entrambi supportano filename, TextWriter, XmlWriter.

```
XElement element = new XElement("myXML");
```

```
element.Save("c:\myTemp\myXML.xml");
```



Creare un documento XML

```
XNamespace ns = "http://example.books.com";  
XDocument books = new XDocument(  
    new XElement(ns + "bookstore",  
        new XElement(ns + "book",  
            new XAttribute("ISBN", isbn),  
            new XElement(ns + "title", "ASP.NET Book"),  
            new XElement(ns + "author",  
                new XElement(ns + "first-name", a.FirstName),  
                new XElement(ns + "last-name", a.LastName)  
            )  
        )  
    );  
books.Save("c:\Books.xml");
```



Linq e documenti XML

```
public void Example()
{
    // Seleziona le pagine che hanno come titolo "Computer"
    string title = "Computer";
    var v = from page in _x.Elements("SitePage")
           where title == page.Element("Title").Value
           select page;
    // Seleziona la categoria della prima pagina trovata
    string category = v.First().Element("Category").Value;
    // Conta il numero di elementi con quel valore di categoria
    int count =
    (from p in _x.Elements("SitePage")
     where p.Element("Category").Value
     == category && p.Element("Visibility").Value == "Regular"
     select p).Count();
}
```

```
<?xml version="1.0"?>
<ArrayOfSitePage xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <SitePage>
    <Visibility>Supplementary</Visibility>
    <Title>C# .NET Examples and Resources</Title>
    <Url></Url>
    <Category>None</Category>
  </SitePage>

  <SitePage>
    <Visibility>Regular</Visibility>
    <Title>Word Count Algorithm in C#</Title>
    <Url>Content/Word-Count-Algorithm.aspx</Url>
    <Category>Algorithms</Category>
  </SitePage>
</ArrayOfSitePage>
```



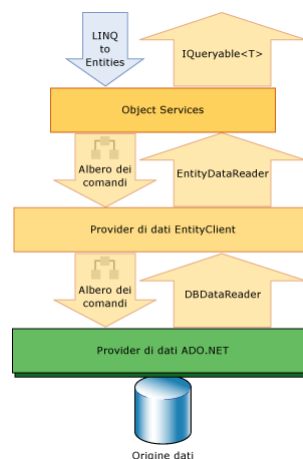
Linq to Entities

La maggior parte delle applicazioni a oggetti si basa su DB relazionali ma i modelli concettuali OO differiscono dal modello relazionale.

Entity Data Model (EDM - Microsoft) è un modello di dati concettuale Entity-Relationships che può essere utilizzato per modellare i dati di un particolare dominio in modo che le applicazioni possano interagire con i dati come entità o oggetti.


Tramite EDM, **ADO.NET** espone le entità come oggetti nell'ambiente .NET. In questo modo, è naturale avvalersi del supporto LINQ.

LINQ to Entities consente agli sviluppatori di scrivere query da eseguire sul DB con lo stesso linguaggio utilizzato per compilare la logica di business.




PLinq

- ✚ **Parallel LINQ (PLINQ)** è un'implementazione in parallelo di LINQ con operatori aggiuntivi per le operazioni in parallelo.
- ✚ PLINQ combina la semplicità e la leggibilità della sintassi di LINQ con la potenza della **programmazione in parallelo**. Il livello di concorrenza è adattato in base alla capacità del computer host.
- ✚ In molti scenari PLINQ può aumentare in modo significativo la velocità d'esecuzione delle query utilizzando in modo più efficiente i core disponibili nel computer host.



The .NET Framework Stack

Il linguaggio Linq 

51


LINQ vs PLINQ

- ✚ **LINQ to Objects query:**

```
int[] output = arr
    .Select(x => IsOdd(x))
    .ToArray();
```

- ✚ **PLINQ query:**

```
int[] output = arr.AsParallel()
    .Select(x => IsOdd(x))
    .ToArray();
```

Il linguaggio Linq 

52

Esempio: array Mapping

```
int[] input = ...
bool[] output = input.AsParallel()
    .Select(x => IsPrime(x))
    .ToArray();
```

input: 1, 6, 3, 8, ..., 2, 7

output: F, F, T, F, ..., T, T

Array to array mapping semplice ed efficiente

Il linguaggio Linq 53

Esempio: sequence Mapping

```
IEnumerable<int> input = Enumerable.Range(1,100);
bool[] output = input.AsParallel()
    .Select(x => IsPrime(x))
    .ToArray();
```

Input Enumerator

Lock

Thread 1, Thread 2, ..., Thread N

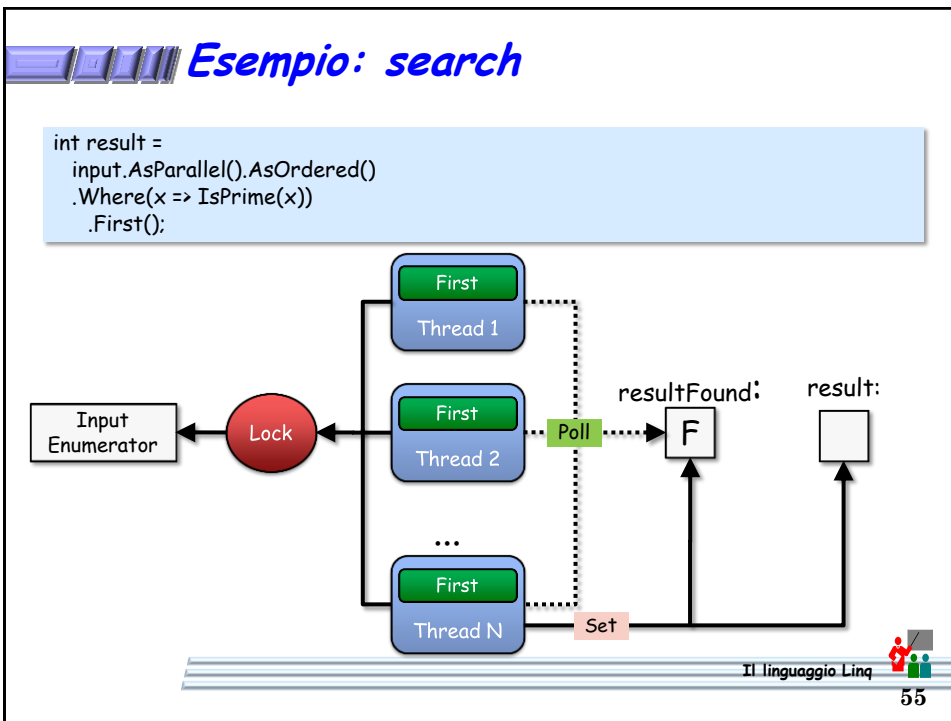
Result s 1, Result s 2, ..., Result s N

output:

I buffer sono ricomposti in un array

Ogni thread elabora una porzione dei dati in input e memorizza i risultati in un buffer.

Il linguaggio Linq 54



Dyrad

- ✦ **Dyrad** è una piattaforma Microsoft in continua evoluzione per applicazioni data parallel.
- ✦ Un'applicazione Dyrad è modellata come un **DAG** (**D**irected **A**cyclic **G**raph).
- ✦ Il DAG definisce il **dataflow** dell'applicazione; i vertici del grafo definiscono le operazioni che sono eseguite sui dati.
- ✦ Dyrad distribuisce il dataflow sui vari nodi di computazione (core in una singola architettura multicore o computer in un cluster),

The diagram shows a person at a computer connected by multiple yellow arrows to a row of server racks, representing data distribution in a cluster.

Il linguaggio Linq 56

