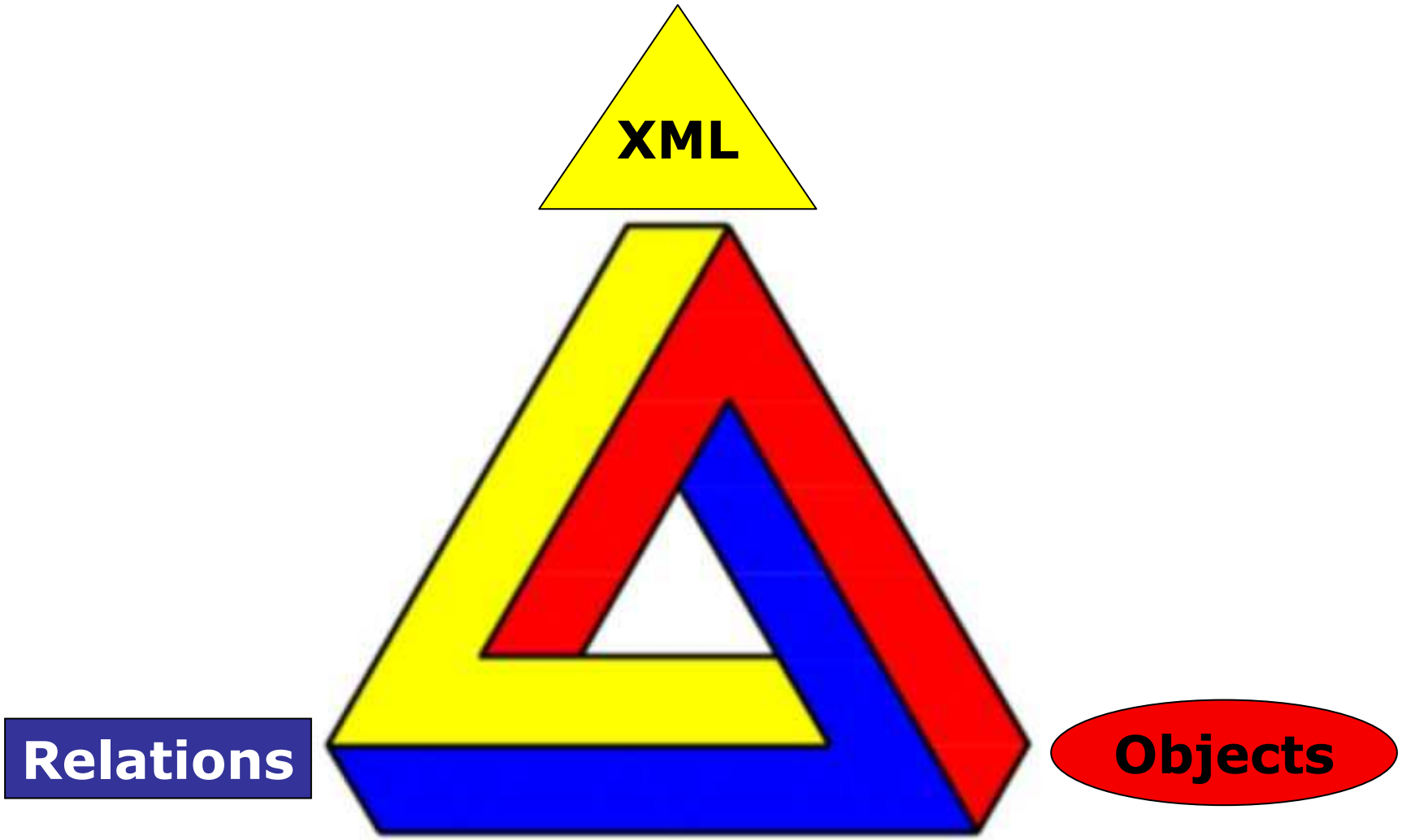
Look at *similarities* rather than differences

The ROX Triangle Of Doom



SQL = data model + query syntax

```
Select Name, Age
From Customers
Where City = "Seattle"
```

Table of rows

XQuery/XPath = data model + query syntax

```
From $C In Customers
Where $C/City = "Seattle"
Return <Cust Name={$C/Name}>
          { $C/Age }
       </Cust>
```

Set of nodes

Objects = data model + query syntax

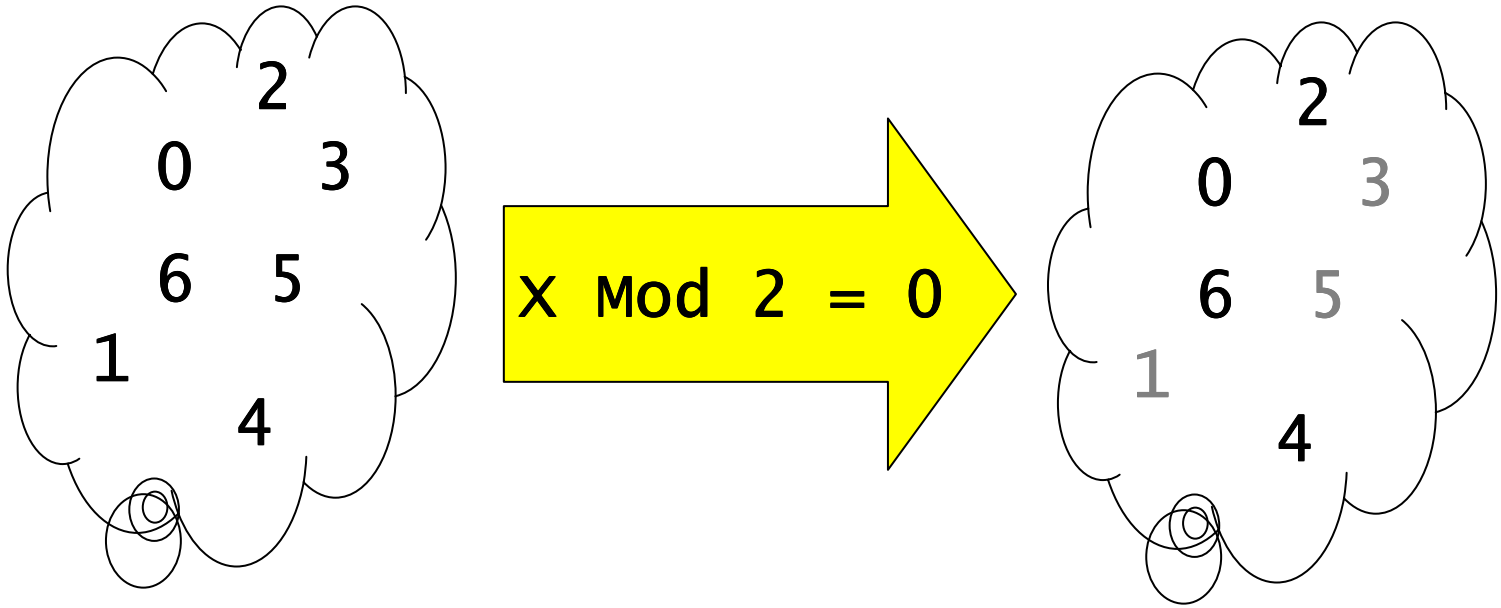
```
Foreach C In Customers
  If C.City = "Seattle"
   R.Add(New With
       {C.Name, C.Age})
  End If
Next
```

Collection of objects

LINQ = Queries Over Arbitrary Collections of Arbitrary Values

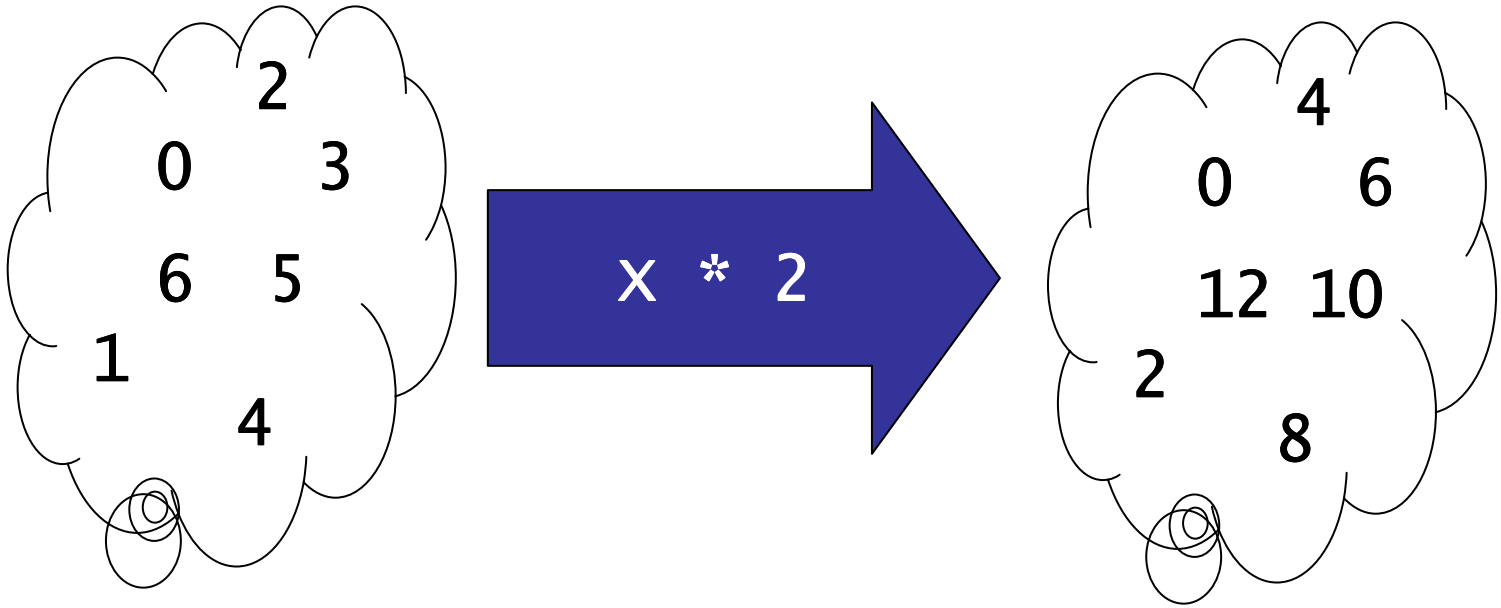
Filtering

T → (T ➔ Bool) → T



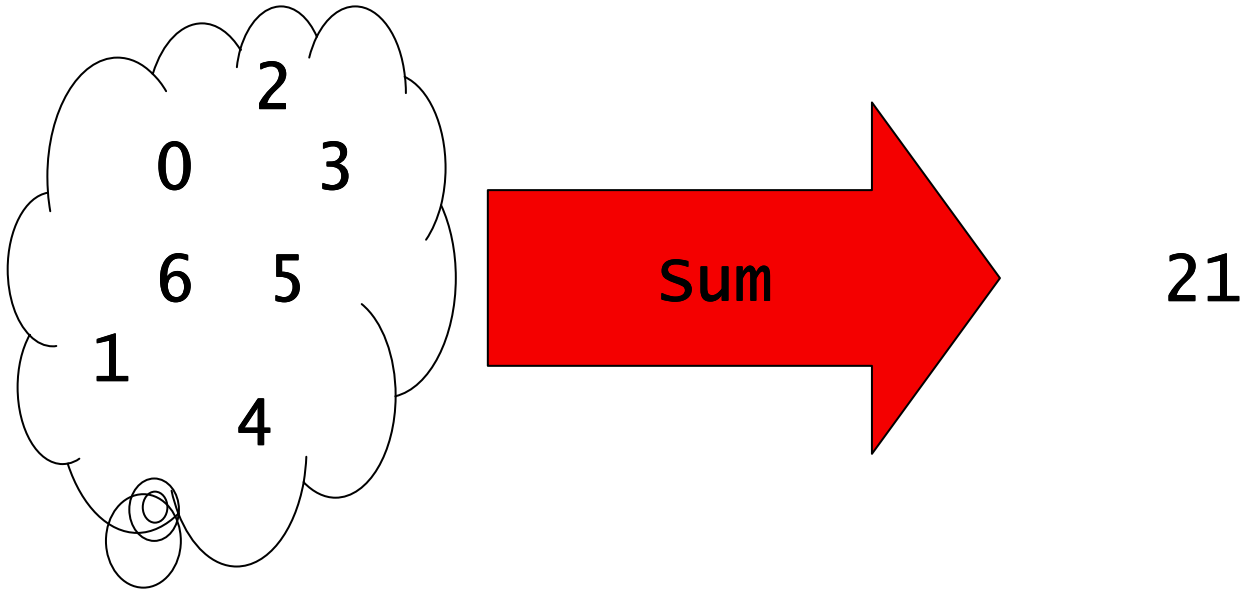
Mapping

T → (T ➔ S) → S



Aggregating

T → (S, (S,T) ➔ S) → S



T → ((T,T) ➔ T) → T

Observation

Many ways to skin a cat

➔

several different choices for "basis" functions.

Need other operations

➔

Sorting, grouping, …

Independent of *collection* type

Independent of *element* type S, T

Independent of *function* type ➔

(mostly)

Standard Sequence Operators

m<T>

IEnumerable<T>
IQueryable<T>

S➔T

Func<S,T>
Expr<Func<S,T>>

m<T> Where<T>
(m<T>src, T➔bool pred)
m<T> SelectMany<S,T>
(m<S> src, S➔m<T> selector)

Standard Query Pattern
(generics not expressive enough)

LINQ == Monads

Formal definition

[edit]

If C is a category, a **monad** on C consists of a functor $T : C \to C$ together with two natural transformations: $\eta : 1_C \to T$ (where 1_C denotes the identity functor on C) and $\mu : T^2 \to T$ (where T^2 is the functor $T \circ T$ from C to C). These are required to fulfill the following conditions (sometimes called coherence conditions):

- $\mu \circ T\mu = \mu \circ \mu T$ (as natural transformations $T^3 \to T$);
- $\mu \circ T\eta = \mu \circ \eta T = 1_T$ (as natural transformations $T \to T$; here 1_T denotes the identity transformation from T to T).

With commutative diagrams:

$$\begin{tikzcd} T^3 \arrow[r, "T\mu"] \arrow[d, "\mu T"'] & T^2 \arrow[d, "\mu"] \\ T^2 \arrow[r, "\mu"'] & T \end{tikzcd} \qquad \begin{tikzcd} T \arrow[r, "\eta T"] \arrow[d, "T\eta"'] \arrow[dr, equal] & T^2 \arrow[d, "\mu"] \\ T^2 \arrow[r, "\mu"'] & T \end{tikzcd}$$

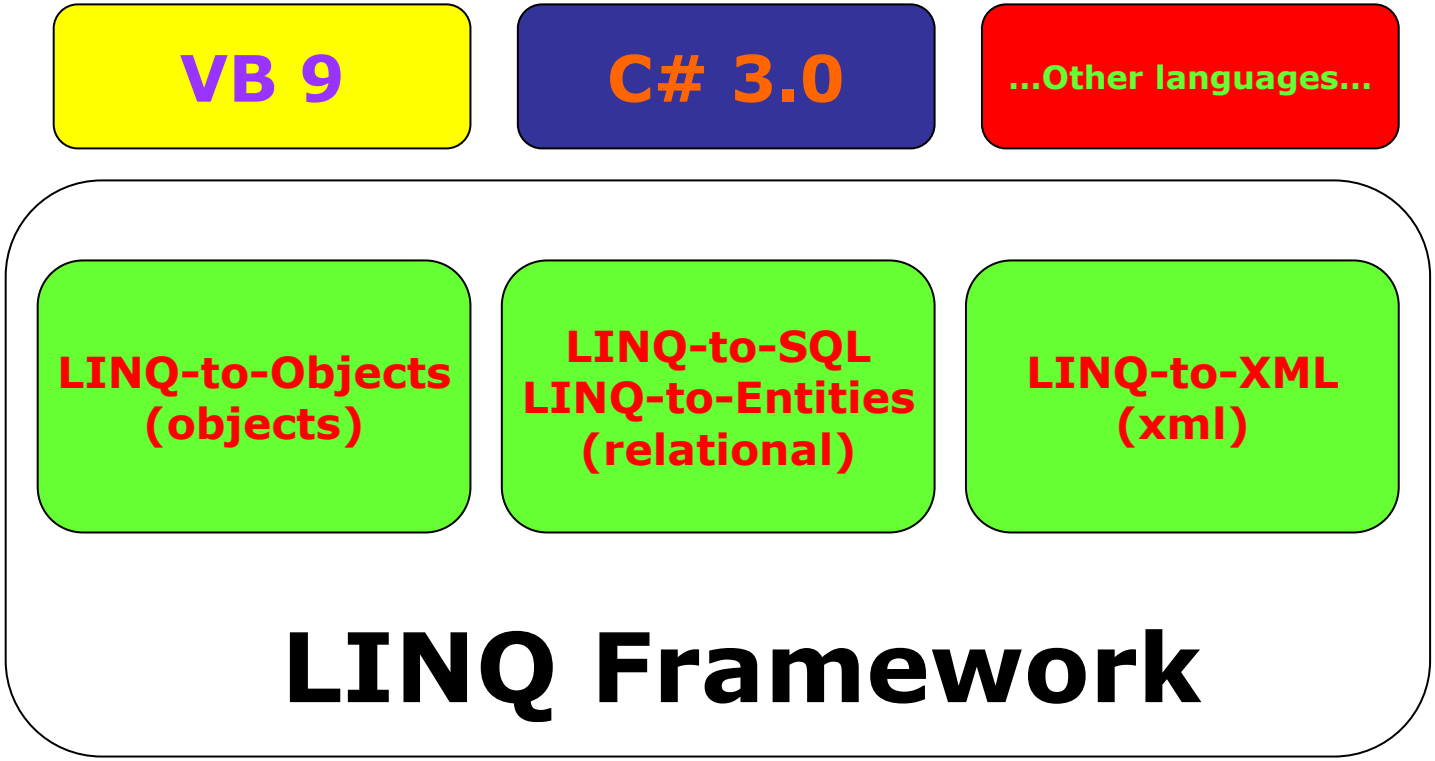
See the article on natural transformations for the explanation of the notations $T\mu$ and μT, or see below the commutative diagrams not using these notions:

$$\begin{tikzcd} T(T(T(X))) \arrow[r, "T(\mu_X)"] \arrow[d, "\mu_{T(X)}"'] & T(T(X)) \arrow[d, "\mu_X"] \\ T(T(X)) \arrow[r, "\mu_X"'] & T(X) \end{tikzcd} \qquad \begin{tikzcd} T(X) \arrow[r, "\eta_{T(X)}"] \arrow[d, "T(\eta_X)"'] \arrow[dr, equal] & T(T(X)) \arrow[d, "\mu_X"] \\ T(T(X)) \arrow[r, "\mu_X"'] & T(X) \end{tikzcd}$$

The first axiom is akin to the associativity in monoids, the second axiom to the existence of an identity element. Indeed, a monad on C can alternatively be defined as a monoid in the category $\mathbf{End_C}$ whose objects are the endofunctors of C and whose morphisms are the natural transformations between them, with the monoidal structure induced by the composition of endofunctors.

LINQ Project

== monad comprehensions in C# & VB



C# 3.0 and Visual Basic 9

Facilitate LINQ

Using general purpose language constructs

Without changing the CLR

Features

- Local Type Inference
- Object & Collection Initializers
- Anonymous Types
- Lambda Expressions
- Query Comprehensions
- Extension Methods
- Expression Trees
- Simplified Properties
- Partial Methods
- Deep XML Support (VB)
- Nullable Types (VB)

Enables Language Extensions via libraries

http://community.bartdesmet.net/blogs/bart/search.aspx?q=LINQ

THE IQUERYABLE TALES - LINQ TO LDAP - PART 1: KEY CONCEPTS

Introduction Welcome to the first real part of our LINQ-to-LDAP series. So far, we've been discussing: Part 0: Introduction In this post, a few key concepts of LINQ concerning the way queries are compiled are discussed. More specifically, we'll talk about IEnumerable<T>, IQueryable<T> and...

Posted Friday, April 06, 2007 8:14 AM by bart

THE IQUERYABLE TALES - LINQ TO LDAP - PART 2: GETTING STARTED WITH IQUERYABLE<T>

Introduction Welcome back to the LINQ-to-LDAP series. So far, we've been discussing: Part 0: Introduction Part 1: Key concepts In the previous post , we put ourselves at the side of the compiler either being faced with an IEnumerable<T> or IQueryable<T> object on which a query is to be performed...

Posted Friday, April 06, 2007 5:45 PM by bart

WHO EVER SAID LINQ PREDICATES NEED TO BE BOOLEAN-VALUED?

Note for purists: This post only speaks for "LINQ predicates", not – although closely related to - the mathematic concept of a predicate as defined by Weisstein, Eric W. "Predicate." From MathWorld --A Wolfram Web Resource. http://mathworld.wolfram.com/Predicate.html as An operator in logic...

Posted Sunday, September 14, 2008 12:23 PM by bart

LINQ THROUGH POWERSHELL

In a reaction to my post on LINQ to MSI yesterday, Hal wrote this: I don't know enough about the dev side to know if this is a stupid question or not but here goes: Would I be able to take advantage of LINQ to MSI (or LINQ in general from a wider point-of-view) from within PowerShell? I know...

Posted Saturday, June 07, 2008 8:44 PM by bart

THE IQUERYABLE TALES - LINQ TO LDAP - PART 4: PARSING AND EXECUTING QUERIES

Introduction Welcome back to the LINQ-to-LDAP series. So far, we've been discussing: Part 0: Introduction Part 1: Key concepts Part 2: Getting started with IQueryable Part 3: Why do we need entities? In the previous post , we entered the domain of implementing a custom query provider for LINQ. More specifically...

Posted Tuesday, April 10, 2007 3:14 PM by bart

UPCOMING EVENT: DEVDAYS 2008 - AMSTERDAM

After my previous little European tour visiting Ghent and Lisboa talking about LINQ, Parallel FX Extensions, Windows PowerShell 2.0 and WPF, I'm looking forward to meet the European audience again at DevDays 2008 Amsterdam . I'm especially thrilled about this one since it's my first speaking...

Posted Friday, April 18, 2008 9:08 PM by bart

THE IQUERYABLE TALES - LINQ TO LDAP - PART 3: WHY DO WE NEED ENTITIES?

Introduction Welcome back to the LINQ-to-LDAP series. So far, we've been discussing: Part 0: Introduction Part 1: Key concepts Part 2: Getting started with IQueryable In the previous post , we discussed quite a bit pieces of the LINQ puzzle, focusing in much detail on expression trees and the role of...

Posted Saturday, April 07, 2007 6:31 PM by bart

LINQ TO MSI - PART 0 - INTRODUCTION

Introduction Lately I've been delivering talks entitled "LINQ to Anything", to be repeated this summer at TechEd Africa. The goal of those talks is to focus on LINQ from the extensibility point of view, in other words: how to write query providers like LINQ to AD or LINQ to SharePoint ...

Posted Friday, June 06, 2008 11:41 PM by bart

LINQ TO SHAREPOINT - IMPROVING THE PARSER == DEBUGGER VISUALIZER FUN

Welcome back to what's going to end up as "LINQ to SharePoint: The Cruel Sequel" :-). The last couple of days, LINQ to SharePoint has been a full-time job and the result is getting better and better build after build. In this post, I'd like to highlight another feature that was planned...

Posted Thursday, July 05, 2007 8:39 PM by bart

VISUAL BASIC 9.0 FEATURE FOCUS - LINQ QUERIES

Welcome back to the Visual Basic 9.0 Feature Focus blog series. In this post, we'll (finally) cover one of .NET Framework 3.5's core features for both C# 3.0 and VB 9.0: LINQ . LINQ stands for Language INtegrated Query and provides querying syntax directly in a general-purpose programming language...

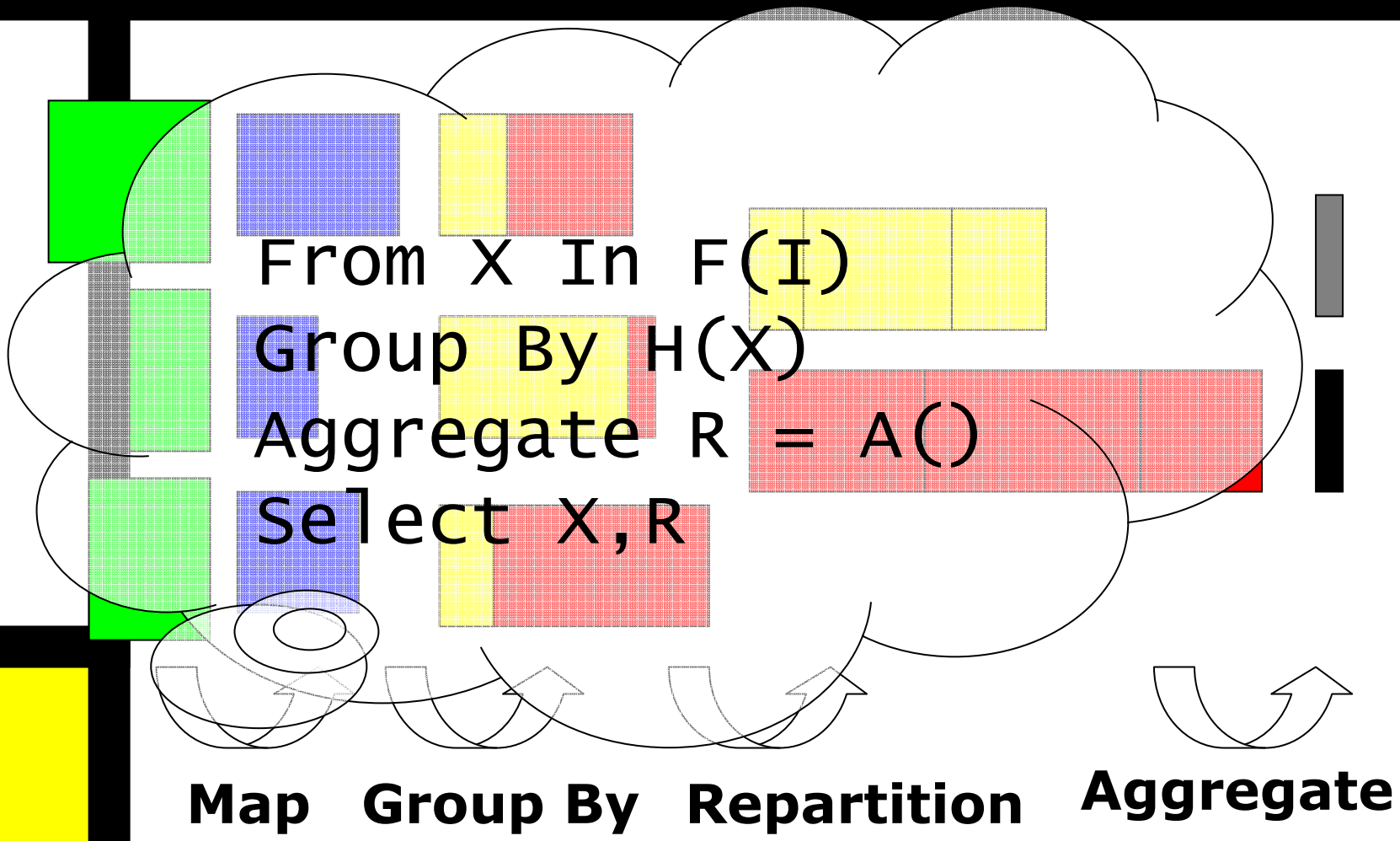
Posted Wednesday, September 05, 2007 12:00 AM by bart

Facets of LINQ

LINQ-*to*-XXX
where
XXX = { SQL, EDM, Sharepoint, …}

Take expression trees
➔
Translate them to
some other query language

LINQ to DataCenter



Sawzall Example 3

```
submitsthroughweek:
      table sum[minute: int] of count: int;

log: P4ChangelistStats = input;

t:  time = log.time; # microseconds

minute: int =
    minuteof(t)
    +60*(hourof(t)
    +24*(dayofweek(t)-1));

emit submitsthroughweek[minute] <- 1;
```

Using Comprehensions

```
var SubmitsThroughWeek =
  from s in db.Submits
  group s by
      s.SubmitTime.Minute
      + 60*(s.SubmitTime.Hour
           + 24*s.SubmitTime.DayOfWeek)
  into g
  select new { minute = g.Key
             , count = g.Count()
             };
```

Facets of LINQ

LINQ-*over*-XXX
where
XXX = { Objects, Simpsons, …}

Implement standard sequence operators
Over some particular types

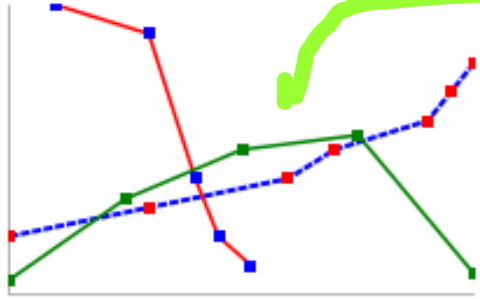
delegate System.Func<T,TResult>
Encapsulates a method that has one parameter and returns a value of the type specified by the TResult parameter.

T is System.Int32
TResult is System.Int32

Facets of LINQ

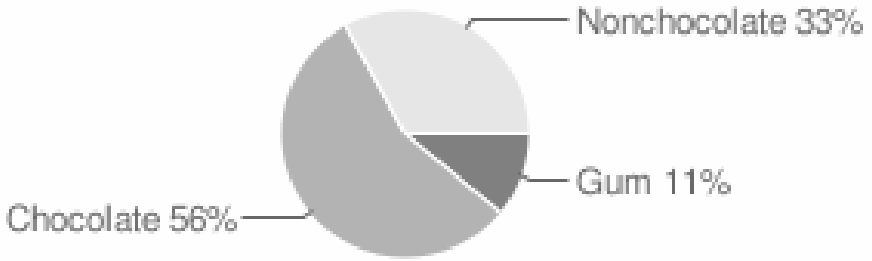
LINQ-*for*-XXX
where
XXX = { XML, Charts, …}

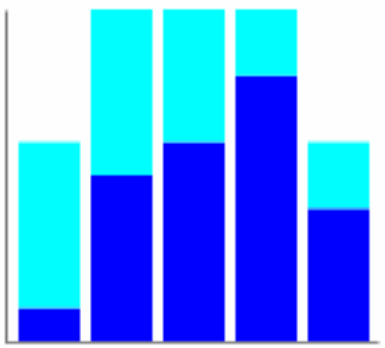
LINQ-friendly API, exposes accessors and constructors as collections



```
var xs0 = new TPoints(new TxT(0, 20), new TxT(30, 30)
                     , new TxT(60, 40), new TxT(70, 50)
                     , new TxT(90, 60), new TxT(95, 70)
                     , new TxT(100, 80)
                     )
                     { Color = Colors.Blue
                     , Marks = Marked.All.Square(Colors.Red, 5)
                     , Style = { Thickness = 2, Segments = 4, Blanks = 1 }
                     };

var xs1 = new TPoints(new TxT(10, 100), new TxT(30, 90)
                     , new TxT(40, 40), new TxT(45, 20)
                     , new TxT(52, 10)
                     ) { Color = Colors.Red
                       , Marks = Marked.All.Square(Colors.Blue, 5)
                       };

var xs2 = new TPoints(5, 33, 50, 55, 7)
               { Color = Colors.Green
               , Marks = Marked.All.Square(Colors.Green, 5)
               };

var line = new LineXY(xs0, xs1, xs2)
         { Size = { Width = 200, Height = 125 } };

ShowChart(line);
```

```
ShowChart(
    new VBar( new T(from d in new[]{1,5,6,8,4}
                        select d*10.0) { Color = Colors.Blue }
            , new T(from d in new[]{1.0,50,2,60,3,100,4,40,5,20,6}
                        where d > 10
                        select d){ Color = Colors.Cyan }
            ){Size = { Width = 200, Height = 125 }}
    );
```





```
ShowChart(
    new Pie( new T(11){ Legend = "Gum+11%", Color = Colors.Gray}
           , new T(56){ Legend = "Chocolate+56%"}
           , new T(33){ Legend = "Nonchocolate+33%"}
           ){Size = { Width = 400, Height = 100 }});
```