

LINQ - LANGUAGE-INTEGRATED QUERY

GV: ThS Phạm Thị Vương

CÁC NGUỒN DỮ LIỆU

- ▶ CSDL Quan hệ
 - ▶ MS SQL, MySQL, IBM DB2, Oracle 10i
 - ▶ Tables, Indexes, Relationships
 - ▶ ADO.NET, Ole, ODBC
- ▶ XML
 - ▶ Text File phân cấp (Hierarchical)
 - ▶ XPath, XML Reader, XmlDocument
- ▶ Object
 - ▶ Array, List, LinkedList, Dictionary...
 - ▶ IEnumerable<T>

XỬ LÝ DỮ LIỆU

- ▶ Phát triển phần mềm hiện đại ngày nay
- ▶ Nhiều nguồn dữ liệu
- ▶ Tầm quan trọng của truy xuất dữ liệu

GIỚI THIỆU

- ▶ Trong 2 thập kỷ trở lại đây kỹ thuật lập trình hướng đối tượng (object-oriented programming) phát triển ổn định.
- ▶ Những hệ thống xử lý qui mô lớn, lượng thông tin khổng lồ.
- ▶ Dữ liệu từ nhiều nguồn khác nhau, ở nhiều dạng khác nhau.
- ▶ Microsoft cho ra đời .NET Language-Integrated Query

KHÓ KHĂN

- Nhiều nguồn dữ liệu lưu trữ
 - Lập trình viên phải học nhiều cách thức truy vấn
- Data Access Code trộn lẫn với Application Code
 - Khó khăn trong kiểm tra, bảo trì
 - Code trở nên phức tạp
- Mô hình dữ liệu
 - Mô hình Data không tương ứng với mô hình Object

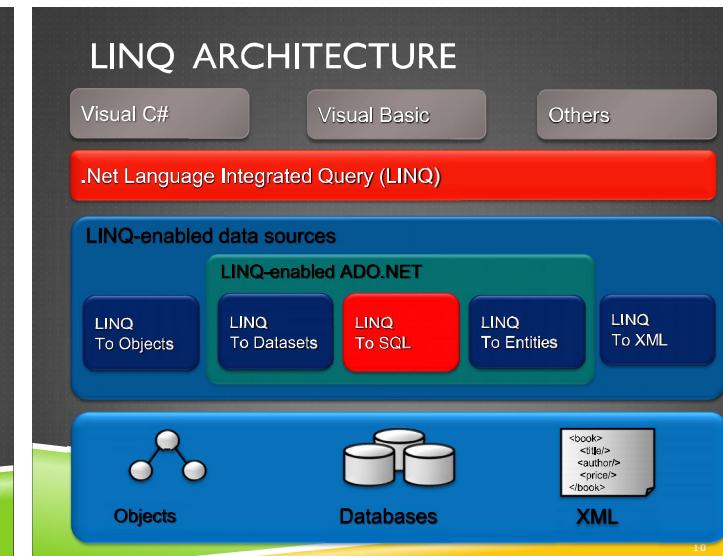
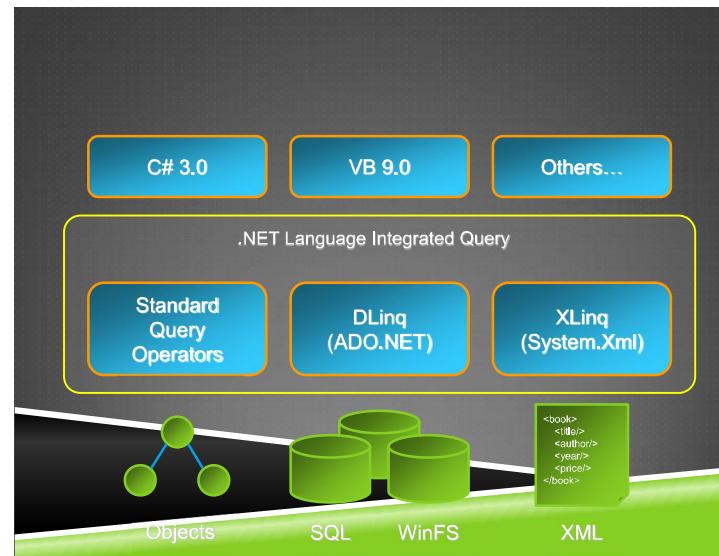
GIỚI THIỆU CHUNG VỀ LINQ

- ▶ Ra đời cùng với Visual Studio 2008 và .NET Framework 3.5
- ▶ Là cầu nối khoảng cách giữa thế giới của các đối tượng với thế giới của dữ liệu.
- ▶ Tích hợp một chuẩn chung cho việc truy xuất dữ liệu từ nhiều nguồn khác nhau.
- ▶ Làm cho việc thao tác dữ liệu dễ dàng hơn.



Anders Hejlsberg
Language INtegrated Query

- Addresses gap (*link – pun intended?*) between programming language and data access
- Brings together objects and data sources
- Provide a solution for object-relational mapping
- A general-purpose querying toolset
- Unifies data access



GIỚI THIỆU CHUNG VỀ LINQ

- **LinQ to Datasets:** Cho phép chúng ta có thể truy vấn các Dataset hoặc DataTable.
- **LinQ to Objects:** Cho phép chúng ta có thể truy vấn các đối tượng trong một tập các phân tử nguồn, các kiểu đối tượng này phải thực thi giao diện IEnumerable hoặc IEnumrable<T>.
- **LinQ to SQL:** Cho phép chúng ta có thể truy vấn các cơ sở dữ liệu quan hệ.
- **LinQ To XML:** Cho phép chúng ta có thể truy vấn các XML Documents.
- **LinQ To Entities:** Cho phép chúng ta có thể truy vấn các thực thể bên trong Entity Framework.

CÂU TRUY VẤN LINQ

► Cú pháp câu truy vấn LinQ

```
from id in source
{ from id in source | where condition }
[ orderby ordering, ordering, ... ]
select expr | group expr by key
[ into id query ]
```

CÁC TOÁN TỬ TRUY VẤN

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>

```

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

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

```

LINQ BENEFITS

- ▶ Intellisense/auto-complete
- ▶ Compile-time syntax checking
- ▶ Can be expanded to new data sources
- ▶ Base implementation can be specialized

LAMDA EXPRESSION

- ▶ Một biểu thức lambda là một hàm ẩn danh có thể chứa các biến và các câu lệnh.
- ▶ Một biểu thức lambda dùng toán tử =>.
- ▶ Ví dụ như biểu thức lambda $x \Rightarrow x * x$ tương đương với

```
int f(int x) { return x * x; }
```

- ▶ Lamda & LinQ :

```
customers.Where(c => c.City == "London");
```

LAMDA EXPRESSION

```

x => x.Length > 0 //input x trả về true nếu x.Length >0 else false
s => s.Length      //input x trả về giá trị x.Length
(x, y) => x == y  //input x,y trả về true nếu x==y else false
(x, y) =>          //input x,y chọn số lớn hơn
{
    if (x > y)
        return (x);
    else
        return (y);
}

```

```

int[] numbers = { 5, 4, 1, 3, 9, 8, 6, 7, 2, 0 };

//Sử dụng Lamda Expression để đếm số lẻ trong một mảng
int oddNumbers = numbers.Count(n => n % 2 == 1);

```

Cú pháp truy vấn dấu chấm (dot)

Dạng này kết hợp một cú pháp biểu thức truy vấn năm trong dấu ngoặc đơn, tiếp theo bởi các toán tử sử dụng cú pháp ký hiệu dấu chấm.

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

var result = (from n in nums
              where n < 5
              orderby n
              select n).Distinct();
```

LINQ TO OBJECT

Sử dụng các Operators:

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

int result = nums.Sum();

Console.WriteLine(result);

Output:27
```

OBJECT RELATIONAL DESIGNER

- ▶ Cung cấp một hình ảnh thiết kế trực quan cho việc tạo LINQ to SQL để tổ chức các thực thể và sự kết hợp (các mối quan hệ) dựa trên các đối tượng trong một cơ sở dữ liệu.
- ▶ O / R Designer cũng cung cấp tính năng cho việc ánh xạ các thủ tục lưu trữ và các hàm để thực hiện các phương thức
- ▶ O / R Designer cung cấp khả năng thiết kế thừa kế các mối quan hệ giữa các lớp thực thể

Nhóm các số cùng số dư khi chia cho 5

```
static void linq_groupby()
{
    int[] numbers = { 5, 4, 1, 3, 9, 8, 6, 7, 2, 0 };

    var numberGroups =
        from n in numbers
        group n by n % 5 into g
        select new { Remainder = g.Key, Numbers = g };

    foreach (var g in numberGroups)
    {
        Console.WriteLine("Numbers with a remainder of {0} when divided
by 5:", g.Remainder);
        foreach (var n in g.Numbers)
        {
            Console.WriteLine(n);
        }
    }
}
```

LINQ TO OBJECT

- ▶ LinQ To Object cho phép chúng ta viết các câu truy vấn trên tập hợp các đối tượng (collections of objects).

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

var result = from n in nums
             where n < 5
             orderby n
             select n;

foreach(int i in result)
    Console.WriteLine(i);
```

Ví dụ demo về LinQ toObject

- LINQ và File Directories: cách LINQ có thể được sử dụng để tương tác với hệ thống tập tin.

```
string startFolder = @"c:\program files\Microsoft Visual Studio 9.0";
// Take a snapshot of the file system.
System.IO.DirectoryInfo dir = new System.IO.DirectoryInfo(startFolder);
// This method assumes that the application has discovery permissions
// for all folders under the specified path.
IEnumerable<System.IO.FileInfo> fileList = dir.GetFiles("*.*", System.IO.SearchOption.AllDirectories);
//Create the query
IEnumerable<System.IO.FileInfo> fileQuery =
    from file in fileList where file.Extension == ".txt" orderby file.Name select file;
//Execute the query. This might write out a lot of files!
foreach (System.IO.FileInfo fi in fileQuery)
{
    Console.WriteLine(fi.FullName);
}
// Create and execute a new query by using the previous
// query as a starting point. fileQuery is not
// executed again until the call to Last()
var newestfile =
    (from file in fileQuery orderby file.CreationTime select new { file.FullName, file.CreationTime }).Last();
Console.WriteLine("\r\nThe newest .txt file is {0}. Creation time: {1}",
newestfile.FullName, newestfile.CreationTime);
```

LINQ TO XML

- Cung cấp khả năng chỉnh sửa tài liệu trong bộ nhớ của DOM và cung cấp khả năng sử dụng biểu thức truy vấn LINQ.

```
XElement xml = new XElement("contacts",
    from c in db.Contacts
    orderby c.ContactId
    select new XElement("contact",
        new XAttribute("contactId", c.ContactId),
        new XElement("firstName", c.FirstName),
        new XElement("lastName", c.LastName)))
);
```

LINQ TO XML

Không sử dụng LinQ

```
XmlDocument doc = new XmlDocument();
XmlElement contacts = doc.CreateElement("contacts");
foreach (Customer c in customers)
    if (c.Country == "USA") {
        XmlElement e = doc.CreateElement("contact");
        XmlElement name = doc.CreateElement("name");
        name.InnerText = c.CompanyName;
        e.AppendChild(name);
        XmlElement phone = doc.CreateElement("phone");
        phone.InnerText = c.Phone;
        e.AppendChild(phone);
        contacts.AppendChild(e);
    }
doc.AppendChild(contacts);
```

<contacts>
<contact>
<name>Great Lakes Food</name>
<phone>(503) 555-7123</phone>
</contact>
...
</contacts>

LINQ TO XML

Lập trình với LinQ

```
XElement contacts = new XElement("contacts",
    from c in customers
    where c.Country == "USA"
    select new XElement("contact",
        new XElement("name", c.CompanyName),
        new XElement("phone", c.Phone)
    )
);
```

LINQ TO SQL

- LINQ to SQL là một phiên bản hiện thực hóa của O/RM (object relational mapping).
- Cho phép bạn mô hình hóa một cơ sở dữ liệu dùng các lớp .NET. Sau đó bạn có thể truy vấn cơ sở dữ liệu (CSDL) dùng LINQ.
- LINQ to SQL hỗ trợ đầy đủ transaction, view và các stored procedure (SP).
- Thêm khả năng kiểm tra tính hợp lệ của dữ liệu và các quy tắc vào trong mô hình dữ liệu.

LINQ TO SQL

Không sử dụng LinQ

```
SqlConnection c = new SqlConnection(...);
c.Open();
SqlCommand cmd = new SqlCommand(
    @"SELECT c.Name, c.Phone
    FROM Customers c
    WHERE c.City = @p0";
    cmd.Parameters.AddWithValue("@p0", "London");
    DataReader dr = cmd.ExecuteReader();
    while (dr.Read()) {
        string name = dr.GetString(0);
        string phone = dr.GetString(1);
        DateTime date = dr.GetDateTime(2);
    }
    dr.Close();
```

LINQ TO SQL

Sử dụng LinQ

```
public class Customer { ... }

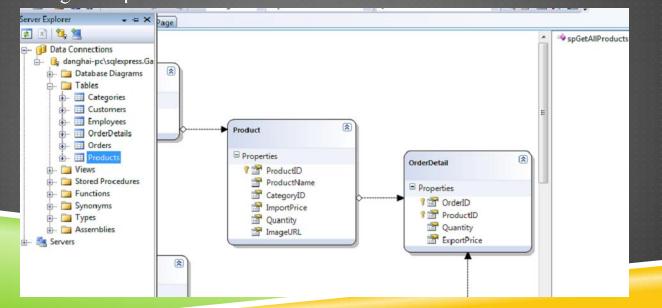
public class Northwind: DbContext
{
    public Table<Customer> Customers;
    ...
}
```

```
Northwind db = new Northwind(...);
var contacts =
    from c in db.Customers
    where c.City == "London"
    select new { c.Name, c.Phone };
```

LINQ TO SQL

► Mô hình hóa CSDL

- ▶ LinQ To SQL Classes(.dbml)
- ▶ Server explorer
- ▶ Drag & drop

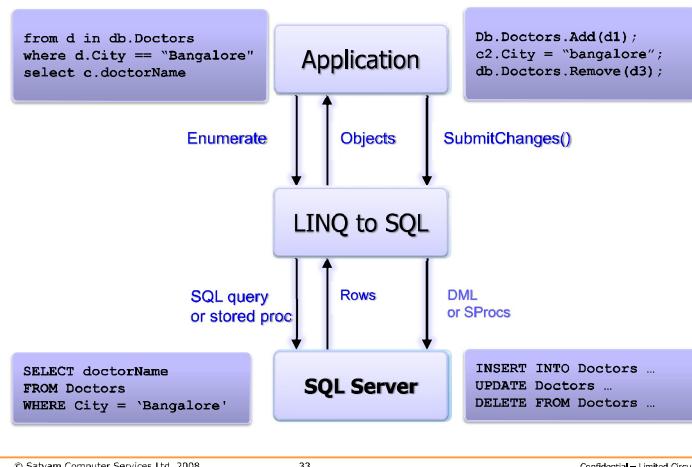


LINQ TO SQL

```
NorthwindDataContext db = new NorthwindDataContext();
var products = from p in db.Products
               where p.Category.CategoryName == "Beverages"
               select new
               {
                   ID = p.ProductID,
                   Name = p.ProductName,
                   NumOrders = p.OrderDetails.Count,
                   Revenue = p.OrderDetails.Sum(o=>o.UnitPrice * o.Quantity)
               };

```

LINQ TO SQL Architecture



LINQ TO SQL

► Truy vấn :

- ▶ Hiện thực hóa lớp `DataContext`.
- ▶ Truy vấn với cú pháp LinQ

```
NorthwindDataContext db = new NorthwindDataContext();

var products = from p in db.Products
               where p.Category.CategoryName == "Beverages"
               select p;
```

LINQ TO SQL

► Cập nhập CSDL:

- ▶ Truy vấn.
- ▶ Thay đổi dữ liệu.
- ▶ Submit.

```
NorthwindDataContext northwind = new NorthwindDataContext();
Product myProduct = northwind.Products.Single(p => p.ProductName == "Chai");
myProduct.UnitPrice = 2;
myProduct.UnitsInStock = 4;
northwind.SubmitChanges();
```

LINQ TO SQL

► Thêm mới vào CSDL:

- ▶ Tạo mới.
- ▶ Gán dữ liệu.
- ▶ `InsertOnSubmit();`
- ▶ `SubmitChanges()`

```
NorthwindDataContext northwind = new NorthwindDataContext();
Product myProduct = new Product();
myProduct.ProductName = "Scott's Special Product";
myProduct.UnitPrice = 999;
myProduct.UnitsInStock = 1;
myProduct.CategoryID = 1;
northwind.Products.InsertOnSubmit(myProduct);
northwind.SubmitChanges();
```

LINQ TO SQL

▶ Xóa trong CSDL:

- ▶ Truy vấn.
- ▶ DeleteAllOnSubmit().
- ▶ Submit.

```
NorthwindDataContext northwind = new NorthwindDataContext();
var lameProducts = from p in northwind.Products
                   where p.OrderDetails.Count > 0 && p.Discontinued == true
                   select p;
northwind.Products.DeleteAllOnSubmit(lameProducts);
northwind.SubmitChanges();
```

LINQ TO SQL

▶ Sử dụng Store Procedure:

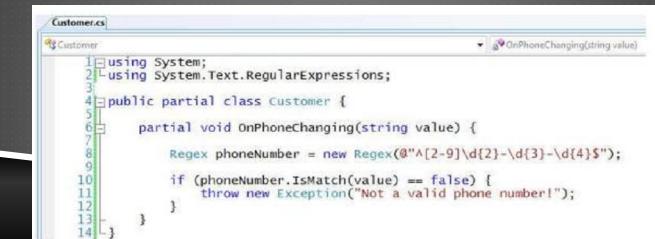
- Mapping.
- Gọi hàm.

```
// Call the stored procedure.
void ReturnRowset()
{
    Northwind db = new Northwind("c:\\northwind.mdf");
    ISingleResult<CustomerByCityResult> result =
    db.CustomersByCity("London");
    foreach (CustomerByCityResult cust in result)
    {
        Console.WriteLine("CustID={0}; City={1}", cust.CustomerID,
        cust.City);
    }
}
```

LINQ TO SQL

▶ Transactions

- ▶ SubmitChanges();
- ▶ Kiểm tra dữ liệu



ENTITY CLASSES

Apply a custom attribute to the top of the class declaration. LINQ to SQL defines the Table attribute for this purpose.

Example:

```
[Table(Name="Customers")]
public class Customer
{
    [Column(Storage=" Cust_Id", DbType="Int NOT NULL"
    , IsPrimaryKey=true)]
    public string CustomerID;
    [Column(Storage=" _City", DbType="VarChar(50) NOT NULL")]
    public string City;
}
```

DATA CONTEXT

- ▶ A channel or platform by which you retrieve objects from the database and resubmit changes
- ▶ It is initialized with a connection or connection string
- ▶ Purpose of the DataContext is to translate your requests for objects into SQL queries made against the database and then assemble objects out of the results.

```
public partial class Northwind : DataContext
{
    public Table<Customer> Customers;
    public Table<Order> Orders;
    public Northwind(string connection): base(connection) {}
}
```

DATA CONTEXT EXAMPLES

// DataContext takes a connection string

```
DataContext db = new DataContext("c:\\northwind\\northwnd.mdf");
```

// Get a typed table to run queries

```
Table<Customer> Customers = db.GetTable<Customer>();
```

var q = from c in Customers where c.City == "London"

select c;

foreach (var cust in q)

```
Console.WriteLine("id = {0}, City = {1}", cust.CustomerID, cust.City);
```

RELATIONSHIPS

- ▶ LINQ to SQL defines an **Association** attribute you can apply to a member used to represent a relationship.
- ▶ An *association relationship* is one like a foreign-key to primary-key relationship that is made by matching column values between tables.

RELATIONSHIPS(MASTER TABLE)

```
[Table(Name="City")]
public class City
{
    [Column(Id=true)]
    public int city_ID;
    ...
    private EntitySet<Doctor> _Doctors;
    [Association(Storage="_Doctors", OtherKey="city_id")]
    public EntitySet<Doctor> Doctors {
        get { return this.Doctors; }
        set { this.Doctors.Assign(value); }
    }
}
```

RELATIONSHIPS (CHILD TABLE)

```
[Table(Name="Doctors")]
public class Doctor {
    [Column(Id=true)]
    public int doctorID;
    [Column]
    public string city_ID;
    private EntityRef<City> _city;
    [Association(Storage="_city", ThisKey="cityID")]
    public Customer Customer {
        get { return this._city.Entity; }
        set { this._city.Entity = value; }
    }
}
```

LINQ QUERY EXAMPLES

```
public class Doctor { ... }

public class DBHelperDataContext: DataContext
{
    public Table<Doctor> Doctors;
    ...
}
```

Classes describe data

Tables are like collections

Strongly typed connections

Integrated query syntax

Strongly typed results

```
DBHelperDataContext db = new
DBHelperDataContext(...);
var contacts =
    from d in db.Doctors
    where d.City == "London"
    select new { d.doctorName };
```

DML OPERATIONS

- Inserting
 - InsertOnSubmit(entity)
 - InsertAllOnSubmit(entityCollection)
 - Constraints are not checked early, queries are always sent to the database.
- Updating
 - SubmitChanges()
- Deleting
 - DeleteOnSubmit(entity)
 - DeleteAllOnSubmit(entityCollection)

LINQ TO SQL

- ▶ Ngôn ngữ tích hợp truy cập dữ liệu
 - ▶ Ánh xạ các table và dữ liệu thành lớp và các đối tượng
 - ▶ Tích hợp sẵn Transactions.
- ▶ Mapping
 - ▶ Các quan hệ được ánh xạ thành thuộc tính
- ▶ Đảm bảo dữ liệu
 - ▶ Tự động theo vết các thay đổi
 - ▶ Kiểm tra dữ liệu.