LINQ

Gang Luo
Xuting Zhao

1

Introduction

- LINQ Architecture

Query Expression

- SQL-like:
  
  ```
  from s in names
  where s.Length == 5
  orderby s
  select s.ToUpper();
  ```

- OO-style:
  
  ```
  names.Where(s => s.Length == 5)
  .OrderBy(s => s)
  .Select(s => s.ToUpper());
  ```

Where, OrderBy, and Select are operators. The arguments to these operators are Lambda Expression.

Lambda Expressions

- Examples:
  
  ```
  s => s.Length == 5
  ```

- Executable function
  
  Anonymous functional. Can be assigned to a delegate variable.
  
  No need to indicate the types.
  
  Can be passed to methods as parameters.

- Expression Tree
  
  Efficient in-memory data representations of lambda expressions
  
  Changing the behaviors of the expressions
  
  Applying your own optimization

Methods Extension

You can control not only by Lambda Expression, but also by methods extension

```csharp
public static class Enumerable {
    public static IEnumerable<T> Where<TSource, TTarget>(
        this IEnumerable<TSource> source,
        Func<TSource, TTarget> predicate) {
        foreach (TSource item in source)
            if (predicate(item))
                yield return item;
    }
}
```

LINQ Operations

- Join
  
  ```
  var query = names.Join(people, n => n, p => p.Name, (n, p) => p);
  ```

  The lambda expression for shaping (n, p) => p will be applied on each matching pairs.
LINQ Operations (cont.)

- **Group Join**
  - The lambda expression for shaping is applied on the outer element and the set of all the inner elements that matches the outer one.
  - Shape the result at a set level

```csharp
var query = names.GroupJoin(people, n => n.p, p => p.Name,
  (n, matching) => new { Name = n.Name, Count = matching.Count() });
```

- **Select Many**
  - Each object in the result set may contain a collection or array.
  - Select many help decompose the structure and flatten the result.

```csharp
var query = names.SelectMany(n => people.Where(p => n.p == p.Name)).
  Select(n => new { Name = n.Name, Count = matching.Count() });
```

- **Aggregation**
  - Standard aggregation operators:
    - Min, Max, Sum, Average.
  - General purpose (generic) operator.

```csharp
Int totalLength = names.Sum(n => n.Length);
```

Spotlight

- `IQueryable<T>` interface will defer the evaluation of the query.
- An expression tree will represent all the deferred queries as a whole.
- Several operations could be “merged”, only one SQL query will be generated and sent to database (Similar to Django).
- Multi-level defer

Spotlight (cont.)

- Nested defer

```csharp
var q = from c in db.Customers
  where c.City == "London"
  select new { c.City, c.Name, c.Phone, c.Address, c.TeamName };
```

- What if you want the intermediate result?

```csharp
List persons = from p in personList
  where p.LastName == "John"
  select p.FirstName;
```

Spotlight (cont.)

- Defer Execution
  - Advantages
    - Performance!
    - Query dependency!
  - Disadvantages
    - Divide one query into multiple ones
    - You iterate the result set 100 times, the query will be executed 100 times.
    - Users have to be very careful

Spotlight (cont.)

- Object of new type could be generated on the fly without first define it.
  - This is useful for projection to select one or more fields of another structure.
  - The type will be dynamically generated with setters and getters to corresponding members. Some common methods is also provided.
  - No other methods will be added to this type. But that is already enough!
  - The object is created and initialized by Anonymous Object Initializer.
LINQ to SQL

- Data Model

```csharp
public class Customer

```

- LINQ to SQL helps connect to relational and manipulate the relational data as objects in memory. It achieves this by translating the operations into SQL statements.

Consistency

- Every object will be tracked by LINQ the moment it is loaded from database.

- The tracking mechanism monitors the manipulation on relationship properties. Once you modify one side of the relationship, LINQ will modify the other to keep it consistent.

- When an object is deleted, it could still exist in memory, but it will not cause inconsistency.

Concurrency

- Optimistic concurrency
- Conflict checking when SubmitChanges() is called
- By default, transaction will abort and an exception will be thrown when a conflict is detected.
- User can handle the conflict in the exception catch block.
- User can set whether or not to detect the conflict when one column gets updated.

Transaction/Update

- When update, first check whether new object is added (by tracking mechanism) if yes, insert statement will be generated. What does Django do here?

- Modification will not hit the database until the SubmitChanges() method is called

- All operations will be translated into SQL statements

Transaction/Update (cont.)

- If an exception is throw during the update, all the changes will be rolled back

- One SubmitChanges() is actually one transaction. (pros and cons?)

- Users can also explicitly indicate a new transaction scope.

LINQ to XML

- The LINQ to XML Class Hierarchy

LINQ to XML

- LINQ to XML
  ```csharp
  var query = from x in xml
  where xSurname == "Surname1"
  select new { Surname = xSurname, FName = xFName, LName = xLName };
  ```

- XML to LINQ
  ```csharp
  var x = from XElement People
  where @People.Name == "Name1"
  select new { People.Name == "People", People.FName == xFName, People.LName == xLName };
  ```

Performance

- LINQ has more control and efficiency in O/R Mapping than NHibernate
  - LINQ: External Mapping or Attribute Mapping
  - NHibernate: External Mapping

- Because of mapping, LINQ is lower than database tools such as SqlDataReader or SqlDataAdapter
  - In large dataset, their performance are more and more similar

Thanks!