

السلام عليكم

اليوم سنكمل مشوارنا مع استعلامات LINQ ولن ادخل بكل تفاصيل الكلمات المحجوزة للاستعلامات وقد أضفت البرنامج الذي تم انشاءه في الدرس الأول بشكل رابط لمن يريد اختصار الوقت والتطبيق مباشرة : [LinqSample](#).

ثانيا : استعلامات Aggregate

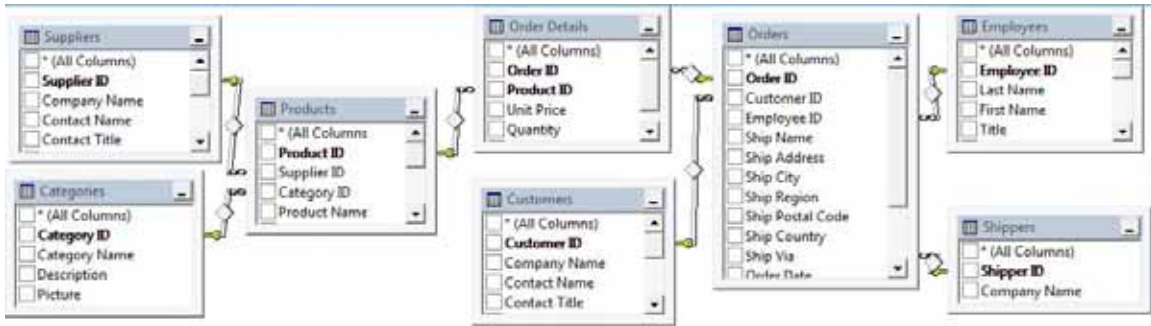
تتميز بانها تعيد قيمة واحدة لأحد دوال التجميع Aggregation **Sum** مجموع القيم – **Average** المعدل الحسابي – **Count** عدد القيم – **Min** أصغر قيمة – **Max** أكبر قيمة . وهي سهلة الاستخدام مثال أضف الكود التالي لبرنامجنا في أي مكان تريد ...

```
Dim CustomerCount = Aggregate cust In MyDB.Customers Into Count()
'SQL : SELECT Count(*) From Customers
Dim OrderSum = Aggregate ord In MyDB.Orders Into Max(ord.OrderDate)
'SQL : SELECT Max([Order Date]) FROM Orders
Console.WriteLine("Custuomers Count: {0} Type : {1} ", CustomerCount,
TypeName(CustomerCount))
Console.WriteLine("Max Order Date Count: {0} Type : {1} ", OrderSum,
TypeName(OrderSum))
```

وقد اضفت جملة SQL الموافقة بعد الاستعلام وكما تلاحظ فإن استعلامات Aggregate تعيد قيمة واحدة دائما ونوع متغير ناتج الاستعلام سيكون أحد انواع DataType وليس مجموعة Collection كما اعتدنا . كما يمكن ان يحتوي استعلام From على تجميع باستخدام Aggregate

العلاقات في LINQ TO SQL :

طبعاً جميعنا يعرف العلاقات وفائدتها . ومع استخدام Linq اصبح لدينا تطبيق سهل ووصول جيد لكافة كائنات قاعدة البيانات ... وكما سبق فإننا نستخدم قاعدة بيانات NorthWind ولذلك قبل الخوض في علاقاتها هذه صورة توضح بنية العلاقات فيها



طبعاً كما ذكرت انشاء العلاقات وحفظها في قاعدة البيانات قبل انشاء الفئات باستخدام sqlMetal يسهل بناء استعلاماتنا بشكل كبير حيث يقوم باني الفئات (sqlmetal أو O/R designer) بتضمين العلاقات وتقديمها لنا بطريقة برمجية ككائنات . مثلاً من الشكل السابق لنفرض أنني اريد معرفة فقط اسماء وارقام المنتجات التي شحنت الى ايطاليا . سيكون عندها استعلام LINQ بهذه البساطة :

```

From det In MyDB.OrderDetails _
Where det.Order.ShipCountry = "Italy" _
Select det.Product.ProductID, det.Product.ProductName Distinct

```

هذا في حالة ترجمة العلاقات في الفئات . أما اذا لم يتم انشاء العلاقة في قاعدة البيانات فانت بحاجة ل Join ويصبح الاستعلام نفسه كالتالي :

```

From det In MyDB.OrderDetails _
Join ord In MyDB.Orders On ord.OrderID Equals det.OrderID _
Join prd In MyDB.Products On prd.ProductID Equals det.ProductID _
Where ord.ShipCountry = "Italy" _
Select prd.ProductID, prd.ProductName Distinct

```

وطبعا نظير الاستعلامين السابقين في SQL هو :

```

SELECT DISTINCT Products.[Product Name] FROM [Order Details]
INNER JOIN Orders ON [Order Details].[Order ID] = Orders.[Order ID]
INNER JOIN Products ON [Order Details].[Product ID] = Products.[Product ID] WHERE (Orders.[Ship Country] = 'Italy'

```

والنتيجة أن علاقة واحد إلى كل أو ما تسمى Foreign Key حيث يكون هناك جدول مفتاح اساسي (-Primary Key) مثل جدول Customers و جدول اخر يحتوي المفتاح الفرعي (Foreign-Key) الذي يشير للمفتاح الاساسي تكون ممثلة في فئات Linq to SQL بشكل :

(كل كائن من جدول Primary-Key له مجموعة من كائنات جدول Foreign-Key)

والعكس غير صحيح

(كل كائن من جدول Foreign-Key ممثل بكائن وحيد من جدول Primary-Key)

وهذه الجمل صحيحة مهما تشعبت العلاقات كما شاهدنا بالمثال السابق.

وهذه الميزة لها عيوب عند التعامل مع كمية كبيرة من البيانات حيث سيتم تحميل الجدول المطلوب مع علاقاته والكائنات الفرعية مما يشكل حملا على الذاكرة والمعالج وهذا يمكن التحكم به من خلال Load Options للكائن DataContext كما سنرى لاحقا .

الاستخدامات الاساسية للاستعلامات :

بناء مصدر بيانات Data Source :

كما رأينا سابقا في المقال الثاني يمكن استخدام أي استعلام Linq كمصدر بيانات لأي عنصر تحكم لعرض البيانات وهناك ملاحظة مهمة بالنسبة لاستعلامات Linq وهي أن أي مجموعة استعلامات تعيد نفس القيمة أو السجل أو نفس مجموعة السجلات تؤثر لنفس الكائن حتى لو كانا استعلامين مختلفين . فقد نتفاجأ عند وضع مصدر بيانات ل combobox و DataGridView من نفس الجدول أنه عند تحريك الاختيار في الكومبو تتحرك المؤشرات في datagrid.

التجميع Grouping :

باستخدام Group By حيث يتم فرز السجلات على شكل مجموعات مستقلة وفقا لمعيار معين لنفرض أنني لا أريد عرض جدول Products بالكامل وانما اريده بطريقة منسقة ومجموعة حسب Category عندها نلجأ ل group By

```

From pro In MyDB.Products _
Group pro By pro.CategoryID Into CatGroup = Group

```

هنا يصبح ناتج الاستعلام عبارة عن مجموعتين بشكل شجري بالتالي نحن بحاجة لحلقتين For Each للتجوال عبر جميع السجلات الحلقة الأولى الأولى تعيد مجموعة CatGroup والحلقة الثانية تتجول في هذه المجموعة كالتالي :

```

Dim mySource = From pro In MyDB.Products _
                Group pro By pro.CategoryID Into CatGroup = Group
For Each grp In mySource
    Console.WriteLine("Category ID = {0}", grp.CategoryID)

```

```

For Each prod In grp.CatGroup
    Console.WriteLine("Product Name: " & prod.ProductName)
Next
Next

```

والاستخدام الاخر الأكثر شيوعا للتجميع هو مع الدوال التجميعية مثلا اذا اردنا أن نجد عدد الطلبات orders لكل موظف Employee سنستخدم الاستعلام التالي :

```

From ord In MyDB.Orders _
Group ord By ord.EmployeeID Into EmpGroup = Group _
Select EmployeeID, OrdersCount = EmpGroup.Count()

```

الترتيب :

باستخدام Order By ويمكن استخدام Take و Skip معها لتعطينا جزء من المجموعة مثلا للحصول على اخر 10 طلبات نستخدم الاستعلام :

```

From ord In MyDB.Orders _
Select ord Order By ord.OrderDate Descending _
Take 10

```

تصفية السجلات :

وهو باستخدام الكلمة Where وبعدها قيمة منطقية و90% من القيم المنطقية والدوال مقبولة بالنسبة Linq to SQL ولاحقا سأعطي أمثلة للدوال واستخدامها بطريقة سهلة مع Linq .

ونصحتي لكل متعلم جديد ل LINQ جرب ثم جرب ثم جرب كل صيغ الاستعلامات الممكنة . لكي تأخذ مرونة في التعامل
 وإلى اللقاء مع المقال القادم الذي سيكون عن تعديل البيانات وازداف وحذف البيانات مع Linq .
 والسلام