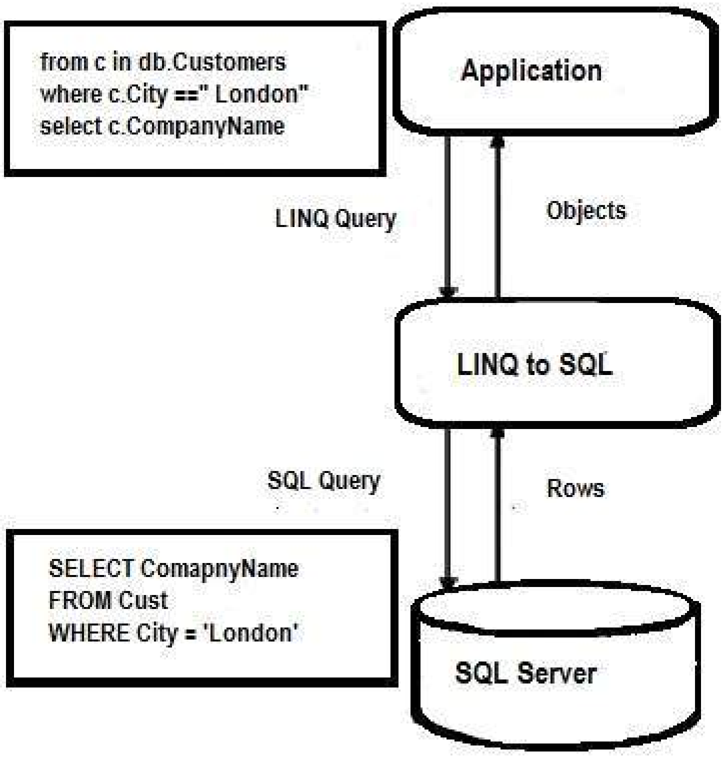
LINQ - SQL

LINQ to SQL offers an infrastructure (run-time) for the management of relational data as objects. It is a component of version 3.5 of the .NET Framework and ably does the translation of language-integrated queries of the object model into SQL. These queries are then sent to the database for the purpose of execution. After obtaining the results from the database, LINQ to SQL again translates them to objects.

Introduction of LINQ To SQL

For most ASP.NET developers, LINQ to SQL (also known as DLINQ) is an electrifying part of Language Integrated Query as this allows querying data in SQL server database by using usual LINQ expressions. It also allows to update, delete, and insert data, but the only drawback from which it suffers is its limitation to the SQL server database. However, there are many benefits of LINQ to SQL over ADO.NET like reduced complexity, few lines of coding and many more.

Below is a diagram showing the execution architecture of LINQ to SQL.

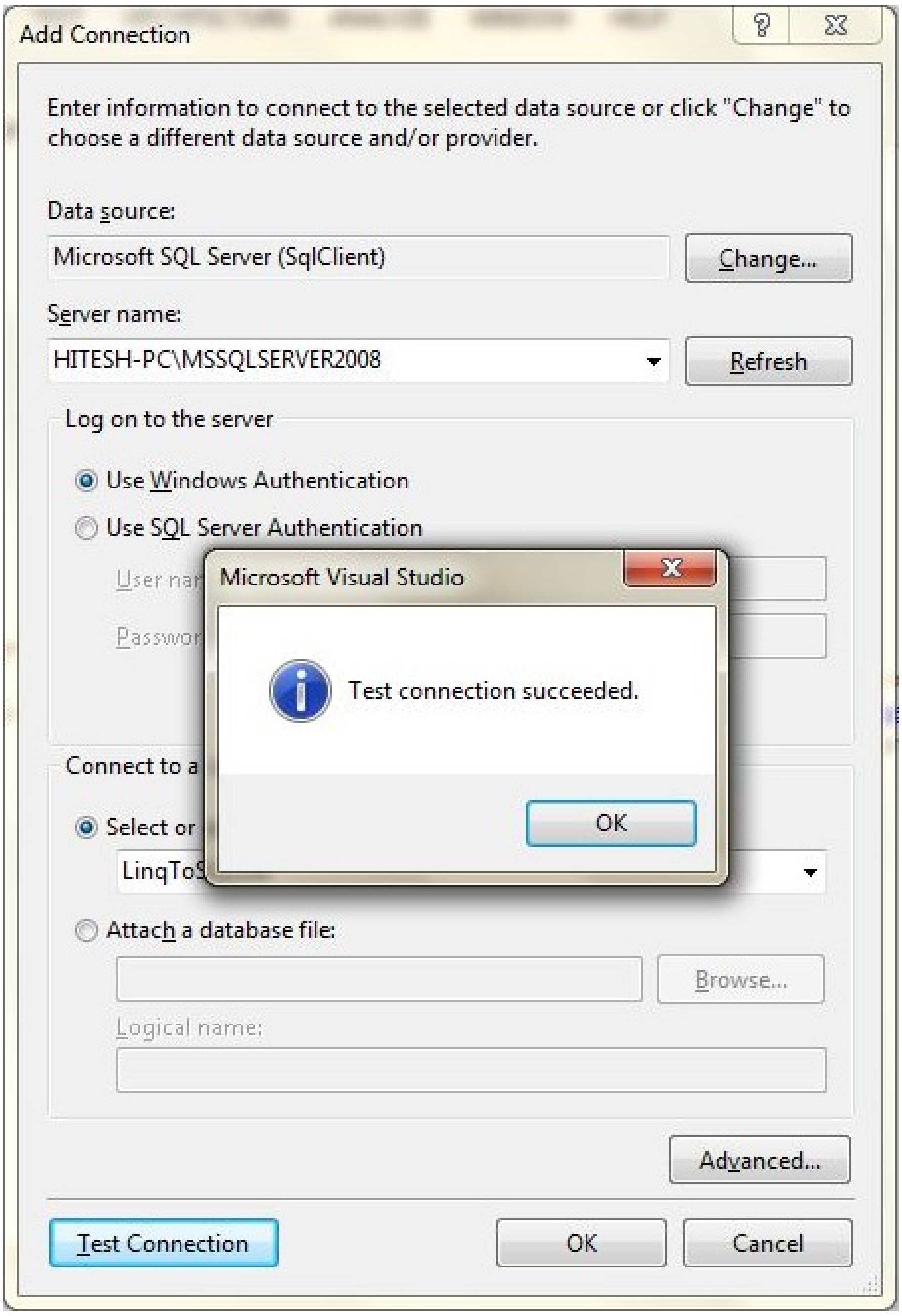


```
from c in db.Customers
where c.City ==" London"
select c.CompanyName
```

```
SELECT ComapnyName
FROM Cust
WHERE City = 'London'
```

How to Use LINQ to SQL?

Step 1 – Make a new “Data Connection” with database server. View &arrar; Server Explorer &arrar; Data Connections &arrar; Add Connection



Add Connection

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source: Microsoft SQL Server (SqlClient)

Server name: HITESH-PC\MSSQLSERVER2008

Log on to the server

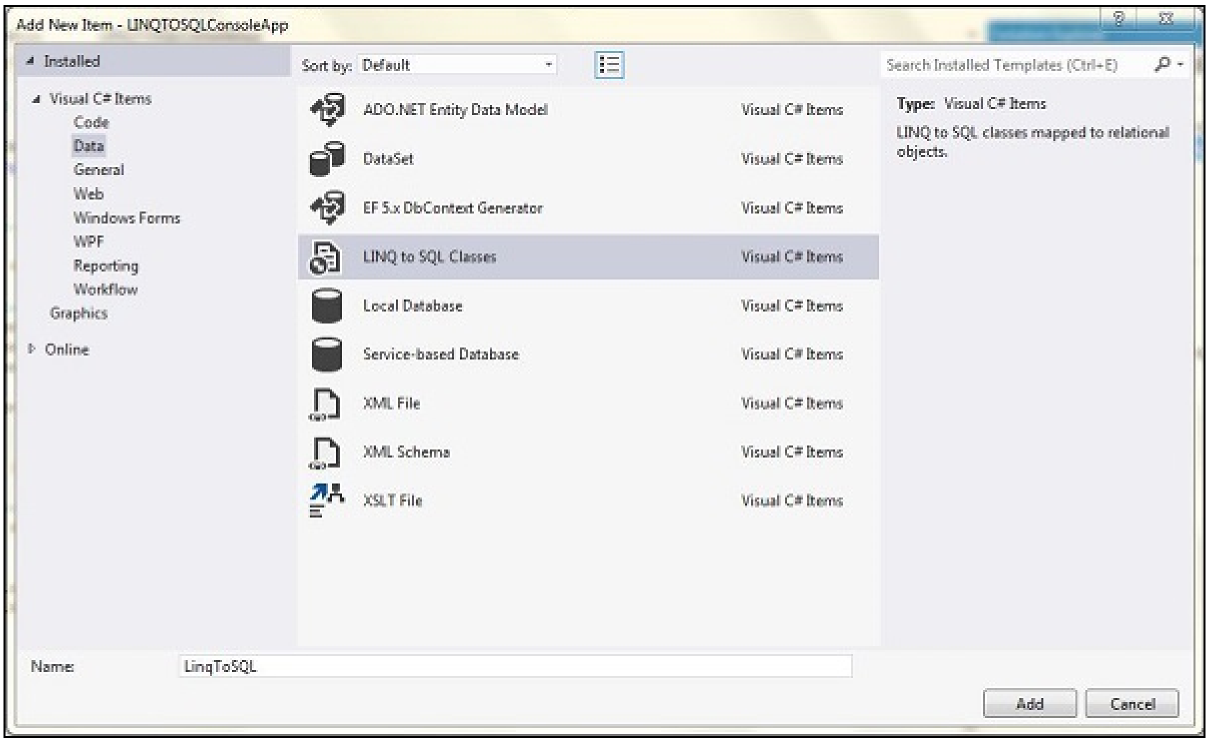
Use Windows Authentication [✓]

Use SQL Server Authentication

Microsoft Visual Studio

Test connection succeeded.

Step 2 – Add LINQ To SQL class file



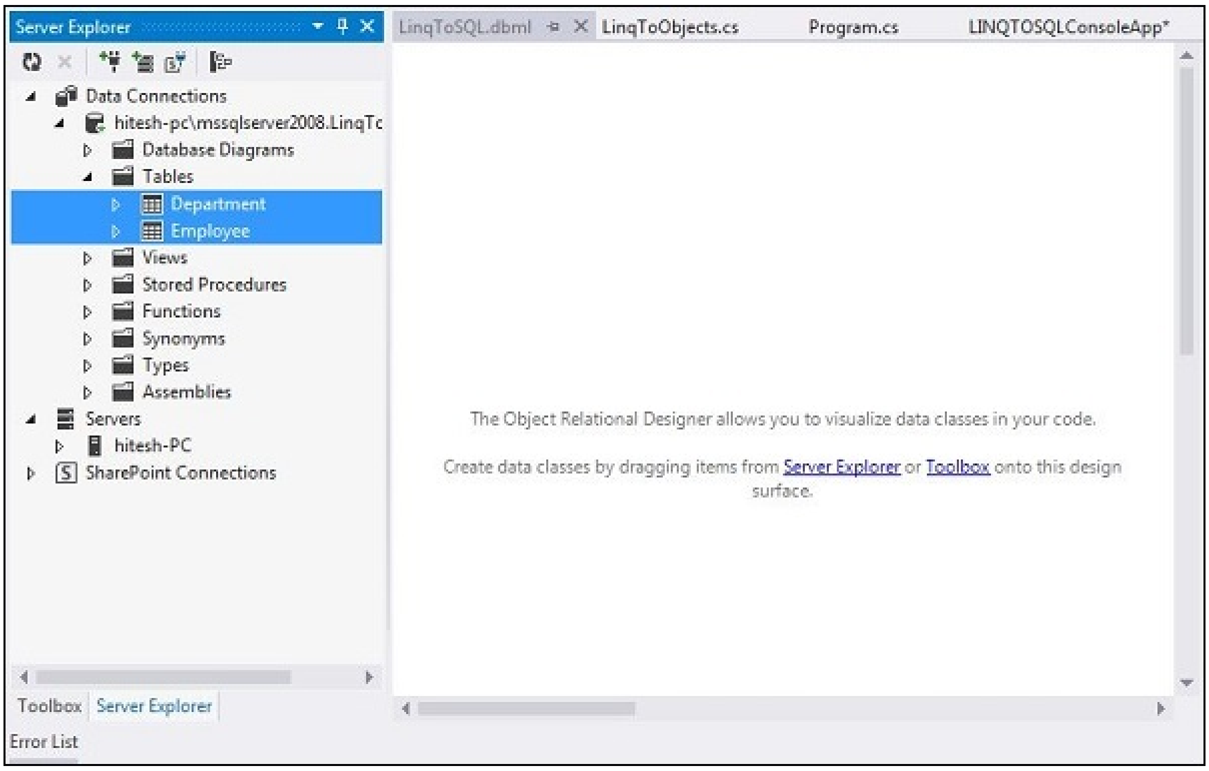
ADO.NET Entity Data Model	Visual C# Items
DataSet	Visual C# Items
EF 5.x DbContext Generator	Visual C# Items
LINQ to SQL Classes	Visual C# Items
Local Database	Visual C# Items
Service-based Database	Visual C# Items
XML File	Visual C# Items
XML Schema	Visual C# Items
XSLT File	Visual C# Items

Type: Visual C# Items

LINQ to SQL classes mapped to relational objects.

Name: LinqToSQL

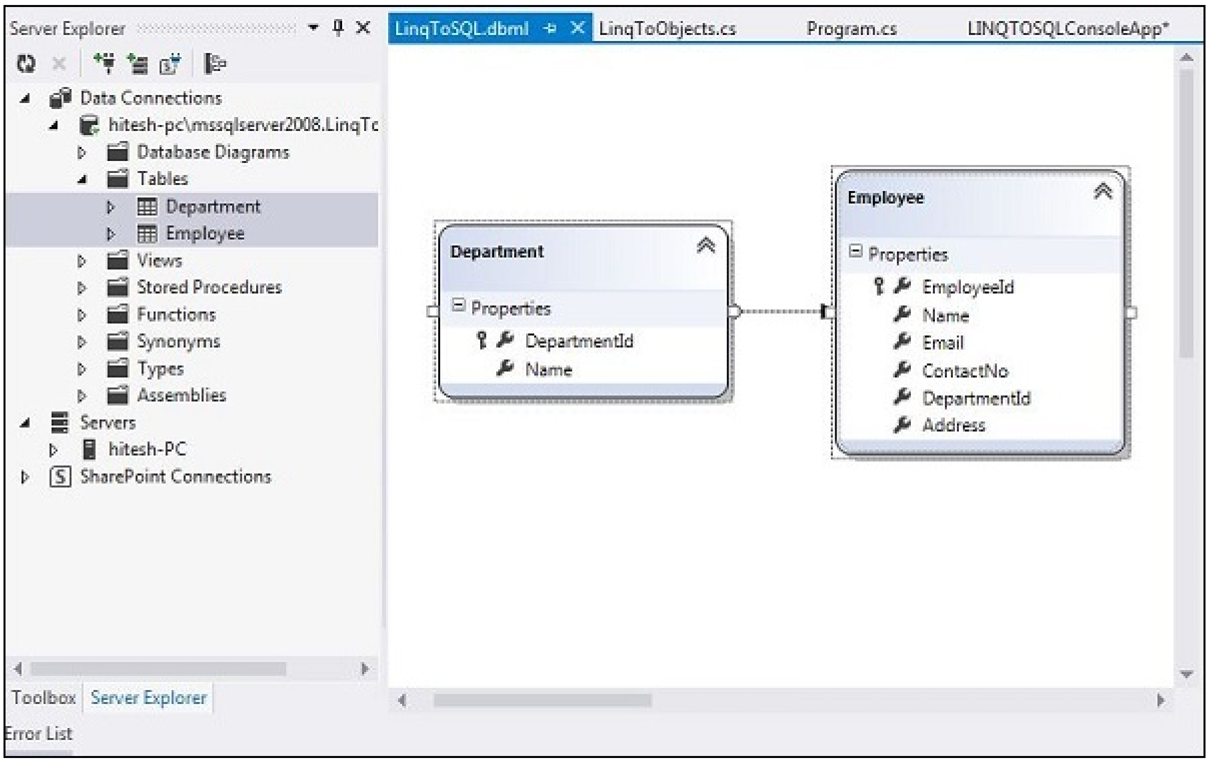
Step 3 – Select tables from database and drag and drop into the new LINQ to SQL class file.



The Object Relational Designer allows you to visualize data classes in your code.

Create data classes by dragging items from Server Explorer or Toolbox onto this design surface.

Step 4 – Added tables to class file.



Querying with LINQ to SQL

The rules for executing a query with LINQ to SQL is similar to that of a standard LINQ query i.e. query is executed either deferred or immediate. There are various components that play a role in execution of a query with LINQ to SQL and these are the following ones.

- **LINQ to SQL API** – requests query execution on behalf of an application and sent it to LINQ to SQL Provider.
- **LINQ to SQL Provider** – converts query to Transact SQL(T-SQL) and sends the new query to the ADO Provider for execution.
- **ADO Provider** – After execution of the query, send the results in the form of a DataReader to LINQ to SQL Provider which in turn converts it into a form of user object.

It should be noted that before executing a LINQ to SQL query, it is vital to connect to the data source via DataContext class.

Insert, Update and Delete using LINQ To SQL

Add OR Insert

C#

```
using System;
using System.Linq;

namespace LINQtoSQL {
   class LinqToSQLCRUD {
      static void Main(string[] args) {
            string connectString = System.Configuration.ConfigurationManager.Connection

            LinqToSQLDataContext db = new LinqToSQLDataContext(connectString);

            //Create new Employee


            Employee newEmployee = new Employee();
            newEmployee.Name = "Michael";
            newEmployee.Email = "yourname@companyname.com";
            newEmployee.ContactNo = "343434343";
            newEmployee.DepartmentId = 3;
            newEmployee.Address = "Michael - USA";

            //Add new Employee to database
            db.Employees.InsertOnSubmit(newEmployee);

            //Save changes to Database.
            db.SubmitChanges();

            //Get new Inserted Employee
            Employee insertedEmployee = db.Employees.FirstOrDefault(e ⇒e.Name.Equals("M

            Console.WriteLine("Employee Id = {0} , Name = {1}, Email = {2}, ContactNo =
                              insertedEmployee.EmployeeId, insertedEmployee.Name, insert
                              insertedEmployee.ContactNo, insertedEmployee.Address);

            Console.WriteLine("\nPress any key to continue.");
            Console.ReadKey();
         }
      }
   }
```

VB

```
  Module Module1

     Sub Main()

        Dim connectString As String = System.Configuration.ConfigurationManager.Connec

        Dim db As New LinqToSQLDataContext(connectString)

        Dim newEmployee As New Employee()
```

```
      newEmployee.Name = "Michael"
      newEmployee.Email = "yourname@companyname.com"
      newEmployee.ContactNo = "343434343"
      newEmployee.DepartmentId = 3
      newEmployee.Address = "Michael - USA"

      db.Employees.InsertOnSubmit(newEmployee)

      db.SubmitChanges()


      Dim insertedEmployee As Employee = db.Employees.FirstOrDefault(Function(e) e.N

      Console.WriteLine("Employee Id = {0} , Name = {1}, Email = {2}, ContactNo = {3
         Address = {4}", insertedEmployee.EmployeeId, insertedEmployee.Name,
         insertedEmployee.Email, insertedEmployee.ContactNo, insertedEmployee.Addres

      Console.WriteLine(vbLf & "Press any key to continue.")
      Console.ReadKey()

   End Sub

End Module
```

When the above code of C# or VB is compiled and run, it produces the following result –

```
Emplyee ID = 4, Name = Michael, Email = yourname@companyname.com, ContactNo =
343434343, Address = Michael - USA

Press any key to continue.
```

Update

C#

```
using System;
using System.Linq;

namespace LINQtoSQL {
   class LinqToSQLCRUD {
      static void Main(string[] args) {

         string connectString = System.Configuration.ConfigurationManager.Connection
```

```
            LinqToSQLDataContext db = new LinqToSQLDataContext(connectString);

            //Get Employee for update
            Employee employee = db.Employees.FirstOrDefault(e =>e.Name.Equals("Michael"

            employee.Name = "George Michael";
            employee.Email = "yourname@companyname.com";
            employee.ContactNo = "99999999";
            employee.DepartmentId = 2;
            employee.Address = "Michael George - UK";


            //Save changes to Database.
            db.SubmitChanges();

            //Get Updated Employee
            Employee updatedEmployee = db.Employees.FirstOrDefault(e =>e.Name.Equals("Ge

            Console.WriteLine("Employee Id = {0} , Name = {1}, Email = {2}, ContactNo =
                              updatedEmployee.EmployeeId, updatedEmployee.Name, updatedE
                              updatedEmployee.ContactNo, updatedEmployee.Address);

            Console.WriteLine("\nPress any key to continue.");
            Console.ReadKey();
         }
      }
   }
```

VB

```
Module Module1

   Sub Main()

      Dim connectString As String = System.Configuration.ConfigurationManager.Connec

      Dim db As New LinqToSQLDataContext(connectString)

      Dim employee As Employee = db.Employees.FirstOrDefault(Function(e) e.Name.Equa

      employee.Name = "George Michael"
      employee.Email = "yourname@companyname.com"
      employee.ContactNo = "99999999"
      employee.DepartmentId = 2
      employee.Address = "Michael George - UK"
```

```
        db.SubmitChanges()

        Dim updatedEmployee As Employee = db.Employees.FirstOrDefault(Function(e) e.Na

        Console.WriteLine("Employee Id = {0} , Name = {1}, Email = {2}, ContactNo = {3
           Address = {4}", updatedEmployee.EmployeeId, updatedEmployee.Name,
           updatedEmployee.Email, updatedEmployee.ContactNo, updatedEmployee.Address)

        Console.WriteLine(vbLf & "Press any key to continue.")
        Console.ReadKey()

     End Sub

  End Module
```

When the above code of C# or Vb is compiled and run, it produces the following result −

```
Emplyee ID = 4, Name = George Michael, Email = yourname@companyname.com, ContactNo =
999999999, Address = Michael George - UK

Press any key to continue.
```

Delete

C#

```
  using System;
  using System.Linq;

  namespace LINQtoSQL {
     class LinqToSQLCRUD {
        static void Main(string[] args) {

           string connectString = System.Configuration.ConfigurationManager.Connection

           LinqToSQLDataContext db = newLinqToSQLDataContext(connectString);

           //Get Employee to Delete
           Employee deleteEmployee = db.Employees.FirstOrDefault(e ⇒e.Name.Equals("Ge

           //Delete Employee
           db.Employees.DeleteOnSubmit(deleteEmployee);

           //Save changes to Database.
```

```
            db.SubmitChanges();

            //Get All Employee from Database
            var employeeList = db.Employees;
            foreach (Employee employee in employeeList) {
               Console.WriteLine("Employee Id = {0} , Name = {1}, Email = {2}, ContactN
                  employee.EmployeeId, employee.Name, employee.Email, employee.ContactN
            }

            Console.WriteLine("\nPress any key to continue.");
            Console.ReadKey();
         }
      }
   }
```

VB

```
Module Module1

   Sub Main()

      Dim connectString As String = System.Configuration.ConfigurationManager.Connec

      Dim db As New LinqToSQLDataContext(connectString)

      Dim deleteEmployee As Employee = db.Employees.FirstOrDefault(Function(e) e.Nam

      db.Employees.DeleteOnSubmit(deleteEmployee)


      db.SubmitChanges()

      Dim employeeList = db.Employees

      For Each employee As Employee In employeeList
         Console.WriteLine("Employee Id = {0} , Name = {1}, Email = {2}, ContactNo =
            employee.EmployeeId, employee.Name, employee.Email, employee.ContactNo)
      Next

      Console.WriteLine(vbLf & "Press any key to continue.")
      Console.ReadKey()
   End Sub

End Module
```

When the above code of C# or VB is compiled and run, it produces the following result –

```
Emplyee ID = 1, Name = William, Email = abc@gy.co, ContactNo = 999999999
Emplyee ID = 2, Name = Miley, Email = amp@esds.sds, ContactNo = 999999999
Emplyee ID = 3, Name = Benjamin, Email = asdsad@asdsa.dsd, ContactNo =

Press any key to continue.
```