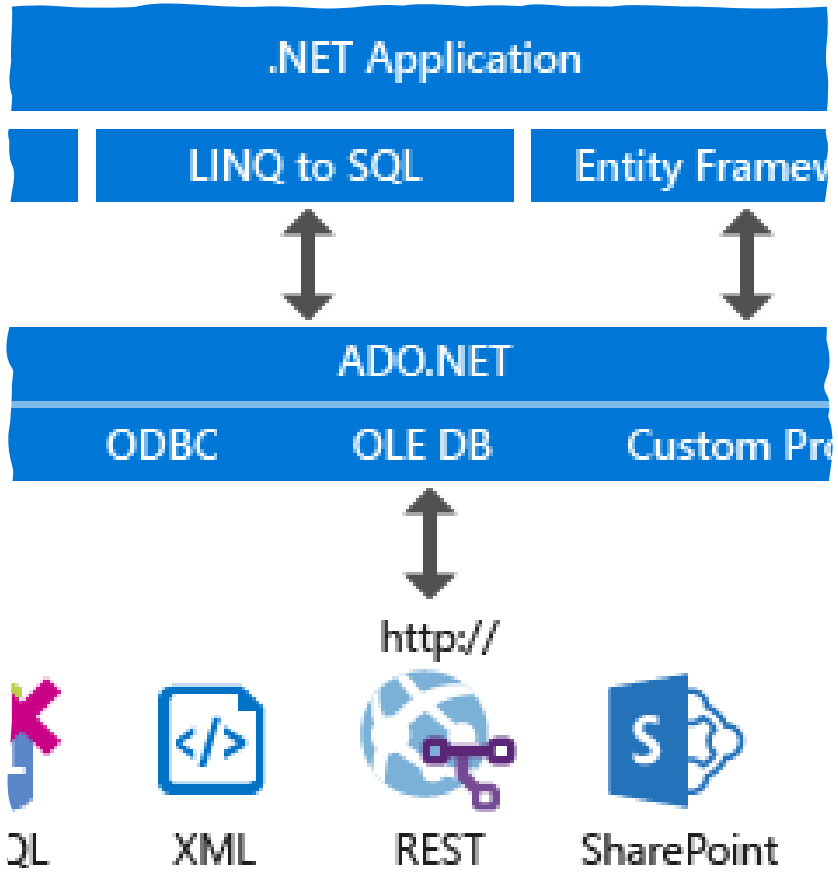
C# and SQL

CRUD (Create, Read, Update, and Delete)

CRUD operations are fundamental in database management and essential for any application that interacts with a database, whether a web application, desktop software, or a mobile app. These operations provide the basic building blocks for managing data throughout its lifecycle in a database system.

- **Create:** This operation involves adding new data records or entries to a database. In the context of a database, it usually means inserting new rows into a table to store new information.
- **Read:** The Read operation involves retrieving or querying existing data from a database. It is used to access and display data stored in the database. In SQL, this is often done using SELECT statements.
- **Update:** Updating refers to modifying or changing existing data in the database. Typically, this is done to update the values of specific fields within a record or row.
- **Delete:** The Delete operation involves removing data or records from the database. It is used to eliminate data that is no longer needed or is obsolete.

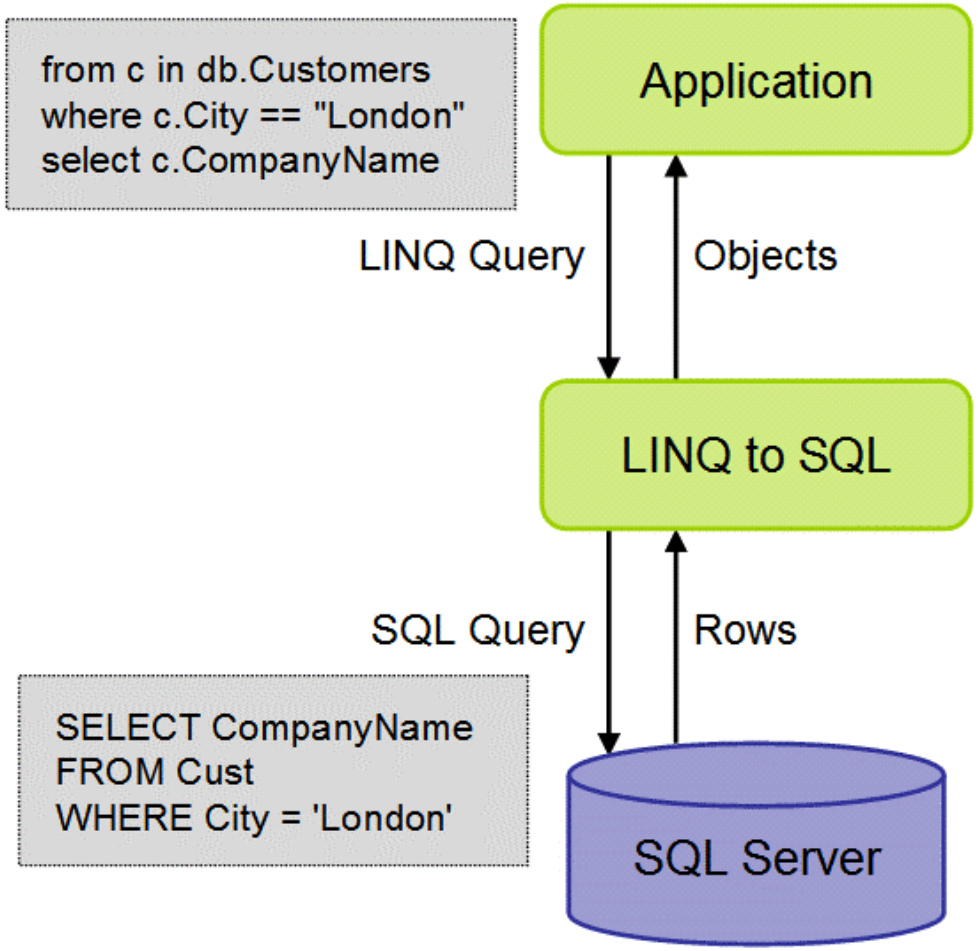
ADO.NET Architecture



- ADO.NET provides consistent access to data sources such as SQL Server and XML, and to data sources exposed through OLE DB and ODBC.
- Data-sharing consumer applications can use ADO.NET to connect to these data sources and retrieve, handle, and update the data that they contain.
- ADO.NET separates data access from data manipulation into discrete components that can be used separately or in tandem.

ADO.NET Architecture LINQ to SQL

- In LINQ to SQL, the data model of a relational database is mapped to an object model expressed in the programming language of the developer.
- When the application runs, LINQ to SQL translates into SQL the language-integrated queries in the object model and sends them to the database for execution.
- When the database returns the results, LINQ to SQL translates them back to objects that you can work within your own programming language.



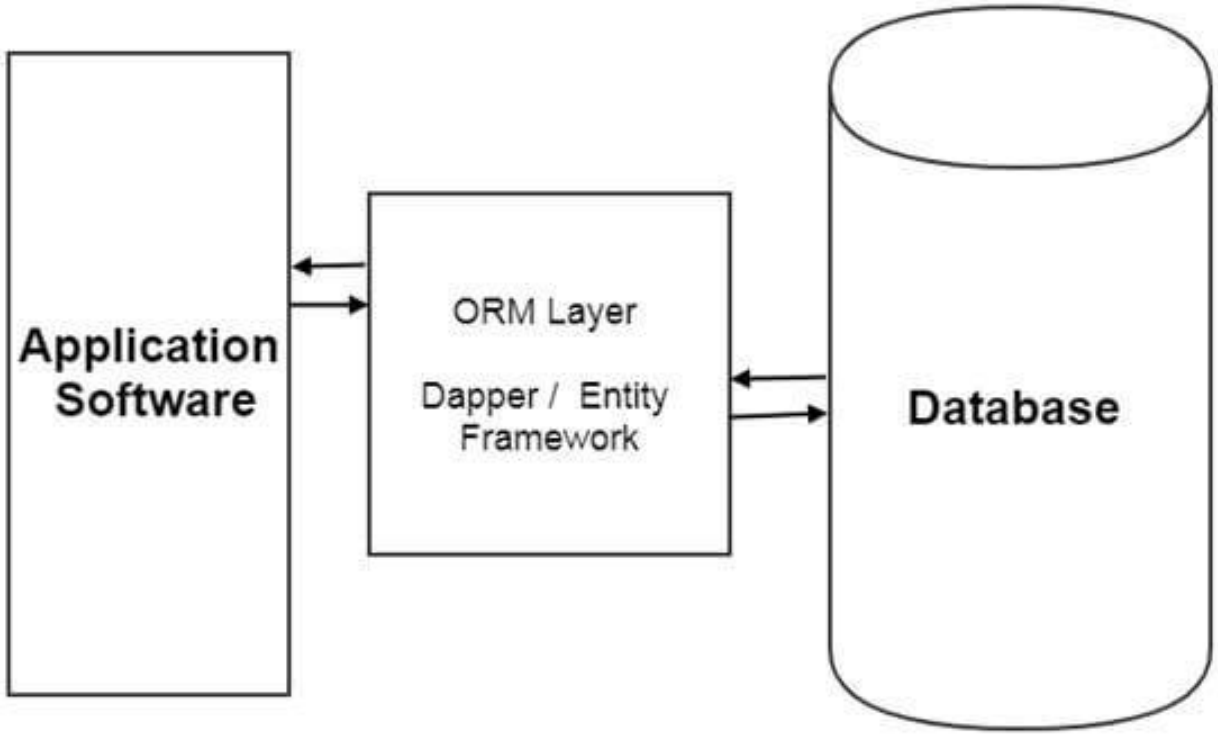
Standard ADO.NET C# code for retrieving data from a database and materialising it as a **collection of Product objects**:

SQL database table

- ProductId – int
- ProductName – varchar
- SupplierId – int
- CategoryId – int
- …

```
var sql = "select * from products";
var products = new List<Product>();
using (var connection = new SqlConnection(connString))
{
    connection.Open();
    using (var command = new SqlCommand(sql, connection))
    {
        using (var reader = command.ExecuteReader())
        {
            var product = new Product
            {
                ProductId = reader.GetInt32(reader.GetOrdinal("ProductId")),
                ProductName = reader.GetString(reader.GetOrdinal("ProductName")),
                SupplierId = reader.GetInt32(reader.GetOrdinal("SupplierId")),
                CategoryId = reader.GetInt32(reader.GetOrdinal("CategoryId")),
                QuantityPerUnit = reader.GetString(reader.GetOrdinal("QuantityPerUnit")),
                UnitPrice = reader.GetDecimal(reader.GetOrdinal("UnitPrice")),
                UnitsInStock = reader.GetInt16(reader.GetOrdinal("UnitsInStock")),
                UnitsOnOrder = reader.GetInt16(reader.GetOrdinal("UnitsOnOrder")),
                ReorderLevel = reader.GetInt16(reader.GetOrdinal("ReorderLevel")),
                Discontinued = reader.GetBoolean(reader.GetOrdinal("Discontinued")),
                DiscontinuedDate = reader.GetDateTime(reader.GetOrdinal("DiscontinuedDate"))
            };
            products.Add(product);
        }
    }
}
```

ORM (Object Relational Mapping Database Tools) Layer



- ORM layer **maps database tables to C# entities and vice versa**.
- We have many ORM libraries available in C#, out of which the below are some main libraries.
  - Entity framework
  - Dapper
  - Nhibernate

https://www.freecodecamp.org/news/what-is-an-orm-the-meaning-of-object-relational-mapping-database-tools/

Dapper

https://dappertutorial.net/

Dapper

- Dapper is a micro-ORM (Object-Relational Mapping) library for .NET.
- It provides a lightweight and efficient way to work with databases, including MSSQL.
- Dapper maps database query results to C# objects.
- It automates the process, reducing the need for manual data mapping code.
- Resulting objects closely resemble your database schema, simplifying development.
- Dapper was written in C# and is a popular choice for data access in C# applications because of its simplicity and efficiency.
- Dapper does not translate queries written in .NET languages to SQL like a full-blown ORM. So, you need to be comfortable writing queries in SQL or have someone write them for you.

https://www.learndapper.com/

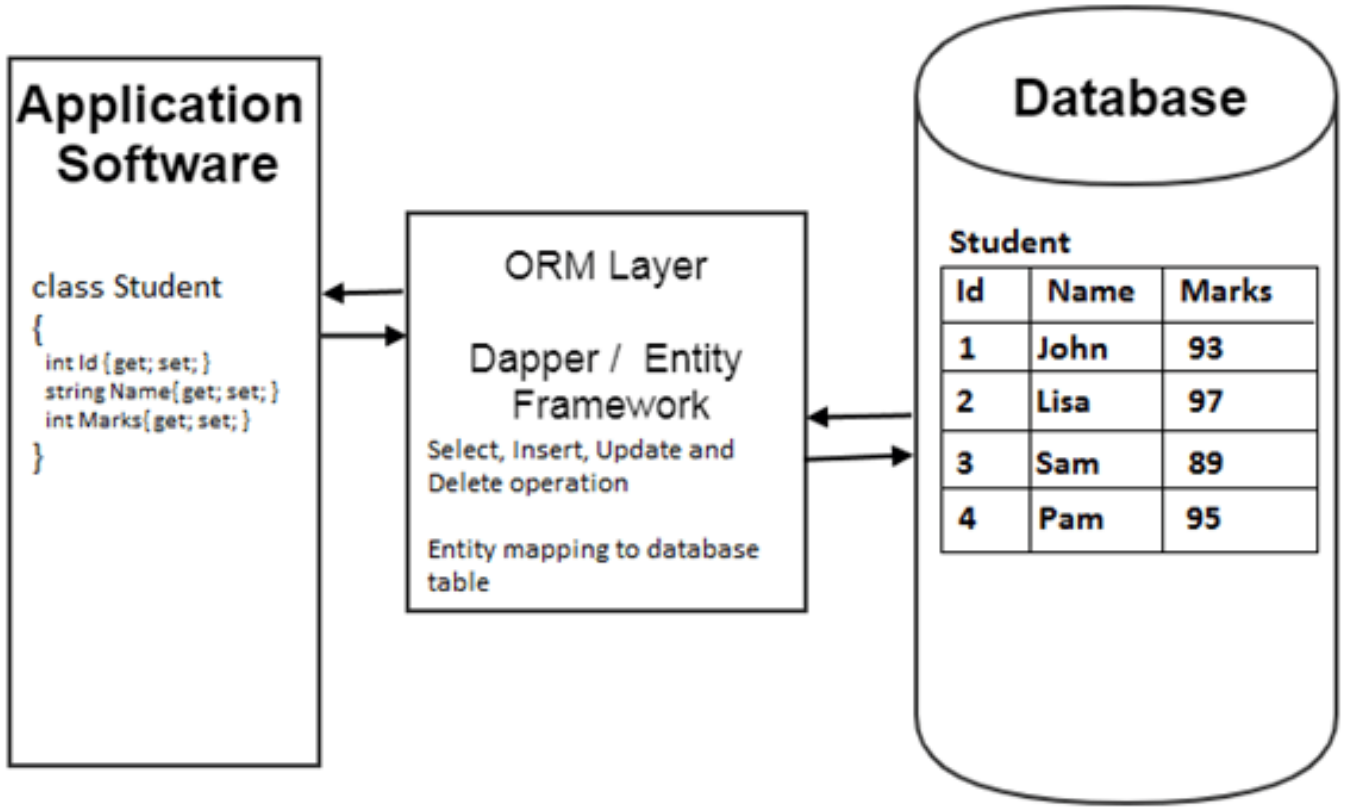
Standard ADO.NET C# code for retrieving data from a database

```
var sql = "select * from products";
var products = new List<Product>();
using (var connection = new SqlConnection(connString))
{
    connection.Open();
products = connection.Query<Product>(sql).ToList();
}
```

```
            var product = new Product
            {
                ProductId = reader.GetInt32(reader.GetOrdinal("ProductId")),
                ProductName = reader.GetString(reader.GetOrdinal("ProductName")),
                SupplierId = reader.GetInt32(reader.GetOrdinal("SupplierId")),
                CategoryId = reader.GetInt32(reader.GetOrdinal("CategoryId")),
                QuantityPerUnit = reader.GetString(reader.GetOrdinal("QuantityPerUnit")),
                UnitPrice = reader.GetDecimal(reader.GetOrdinal("UnitPrice")),
                UnitsInStock = reader.GetInt16(reader.GetOrdinal("UnitsInStock")),
                UnitsOnOrder = reader.GetInt16(reader.GetOrdinal("UnitsOnOrder")),
                ReorderLevel = reader.GetInt16(reader.GetOrdinal("ReorderLevel")),
                Discontinued = reader.GetBoolean(reader.GetOrdinal("Discontinued")),
                DiscontinuedDate = reader.GetDateTime(reader.GetOrdinal("DiscontinuedDate"))
            };
            products.Add(product);
        }
    }
}
```

Dapper methods

- Execute
- Query
- QueryFirstOrDefault
- QuerySingle
- QuerySingleOrDefault
- QueryMultiple



Is Dapper a real ORM?

- **Dapper falls into a family of tools known as micro-ORMs.** These tools perform only a subset of the functionality of full-blown Object Relations Mappers, such as Entity Framework Core, but Dapper is known for its speed and simple implementation compared to others.
- Dapper concentrates its efforts on the **O** and **M** of *ORM* - **O**bject **M**apping.

	Micro ORM	ORM
Map queries to objects	✔	✔
Caching results	✘	✔
Change tracking	✘ [1]	✔
SQL generation	✘ [2]	✔
Identity management	✘	✔
Association management	✘	✔
Lazy loading	✘	✔
Unit of work support	✘	✔
Database migrations	✘	✔

Dapper SQL Server

Using Dapper to query SQL Server is straightforward. The first step is to install Microsoft.Data.SqlClient NuGet package.

```
// Connect to the database
using (var connection = new SqlConnection(connectionString))
{
    // Create a query that retrieves all books with an author name of "John Smith"
    var sql = "SELECT * FROM Books WHERE Author = @authorName";

    // Use the Query method to execute the query and return a list of objects
    var books = connection.Query<Book>(sql, new { authorName = "John Smith"}).ToList();
}
```

Connection

To work with the data in a database, the first obvious step is the connection. The connection to a database normally consists of the below-mentioned parameters.

- **Database name or Data Source** – The first important parameter is the database name to which the connection needs to be established. Each connection can only work with one database at a time.
- **Credentials** – The next important aspect is the username and password which needs to be used to establish a connection to the database. It ensures that the username and password have the necessary privileges to connect to the database.
- **Optional parameters** – For each database type, you can specify optional parameters to provide more information on how .net should handle the connection to the database. For example, one can specify a parameter for how long the connection should stay active. If no operation is performed for a specific period of time, then the parameter would determine if the connection has to be closed.

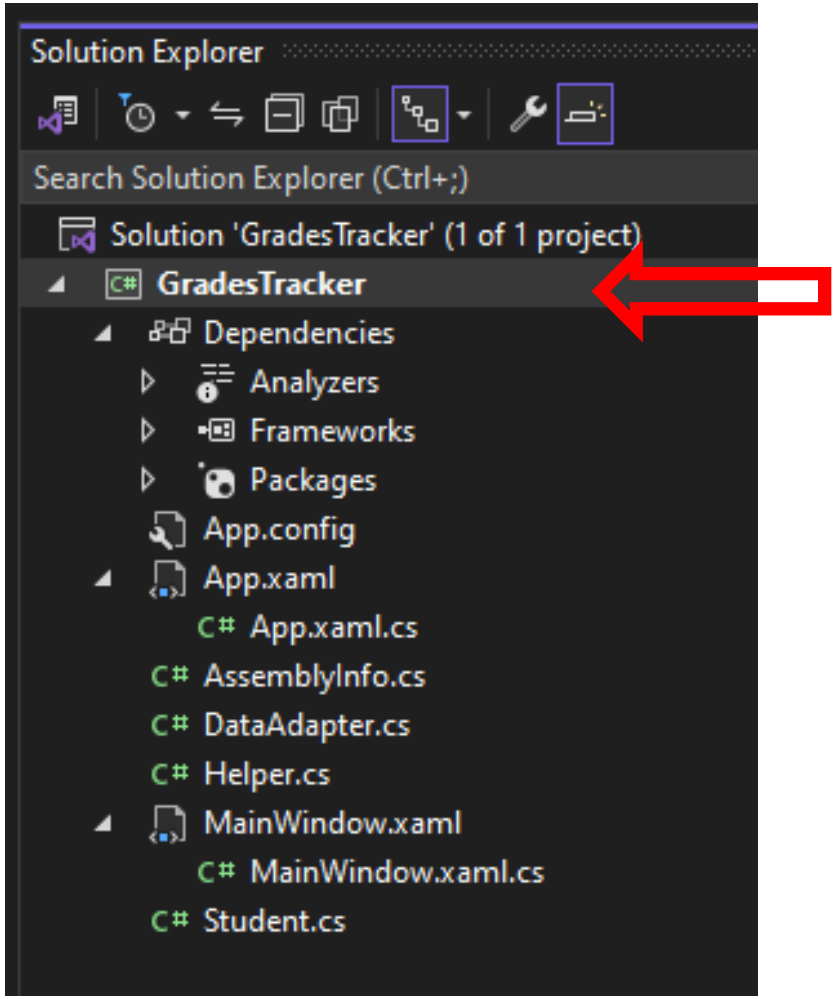
Operations

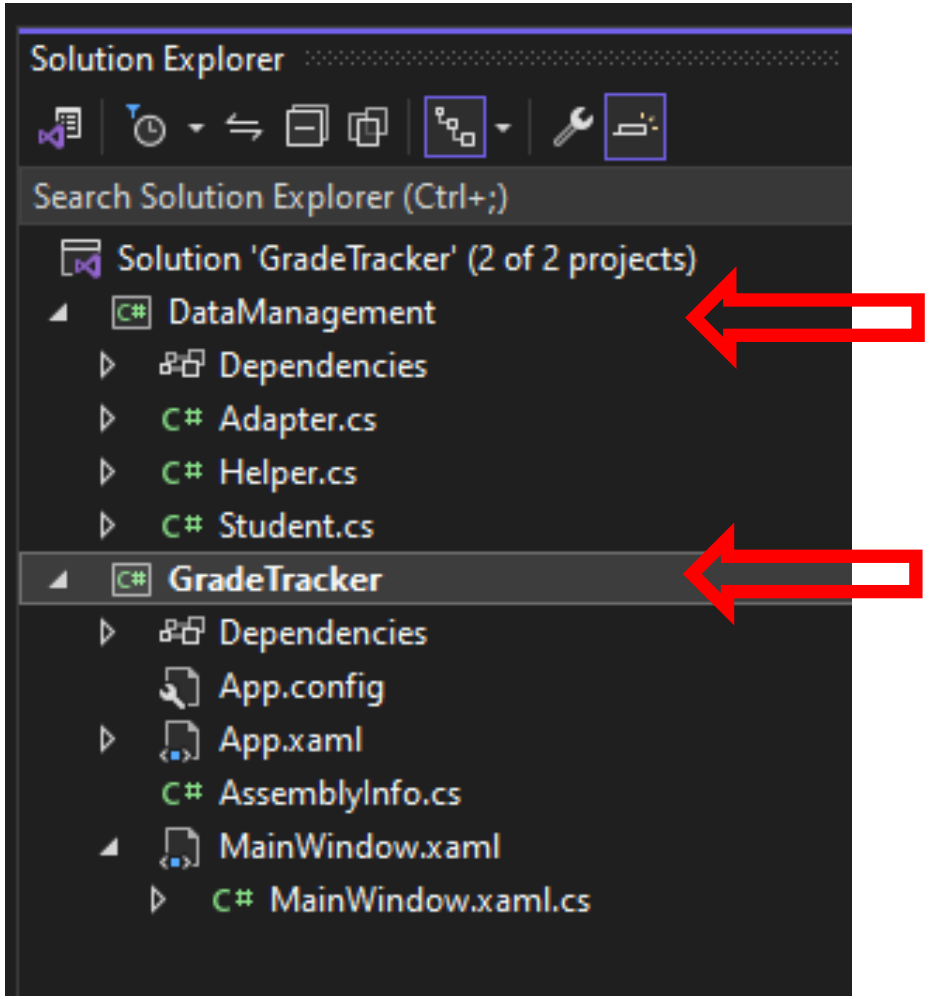
- **Selecting data from the database** – Once the connection has been established, the next important aspect is to fetch the data from the database. C# can execute the ‘SQL’ select command against the database. The ‘SQL’ statement can be used to fetch data from a specific table in the database.
- **Inserting data into the database** – C# can also be used to insert records into the database. Values can be specified in C# for each row that needs to be inserted into the database.
- **Updating data into the database** – C# can also be used to update existing records in the database. New values can be specified in C# for each row that needs to be updated in the database.
- **Deleting data from a database** – C# can also be used to delete records into the database. Select commands to specify which rows need to be deleted can be specified in C#.

Grade Tracker. DAPPER_SQL_WPF guide.docx

1. **App.config** - configuration file to your C# project to store the details of your connection string and allow you to connect to your database.
2. **Data Management Architecture (project)**
   - **Helper class** -to generate the SQL connections when we need them in our other classes
     - **System.Data.SqlClient** library
     - **GetConnectionString()** method - access the **App.config** file and return the connection string
     - **CreateConnection()** method - create a new **SQLConnection**
   - **Class to map sql table**
   - **DataAdapter class** - to handle our data requests to and from our SQL database
3. **Dapper** - Object-Relational Mapping library

Solution structure

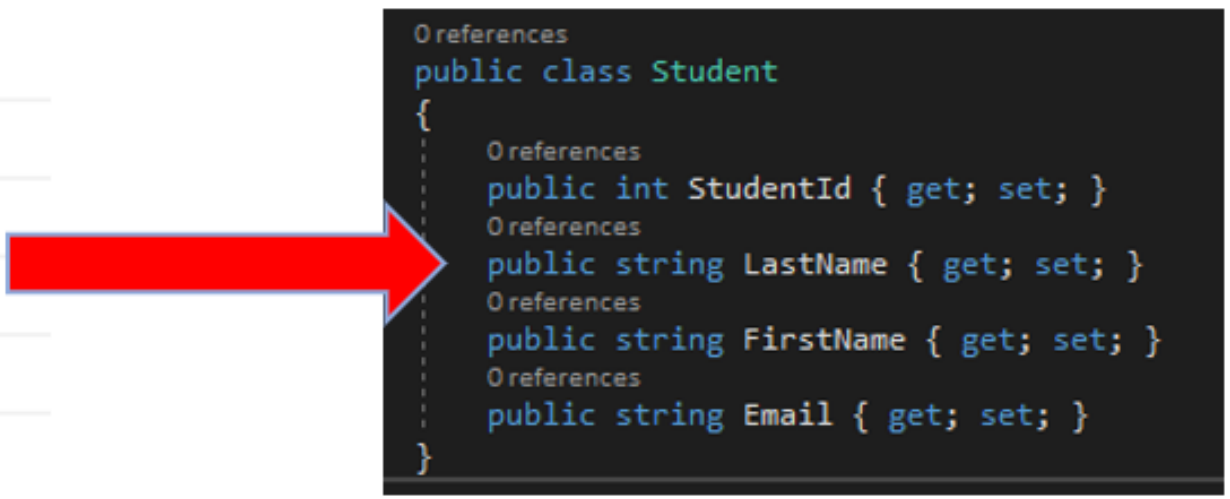




Common SQL data types and their C# equivalents

SQL Data Type	C# Data Type
int	int
decimal	decimal
bit	boolean
varchar	string
DateTime	DateTime

StudentId	int
LastName	varchar(50)
FirstName	varchar(50)
Email	varchar(50)

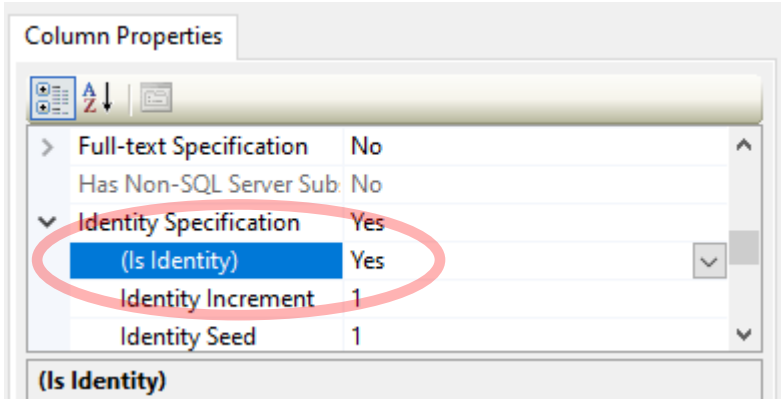


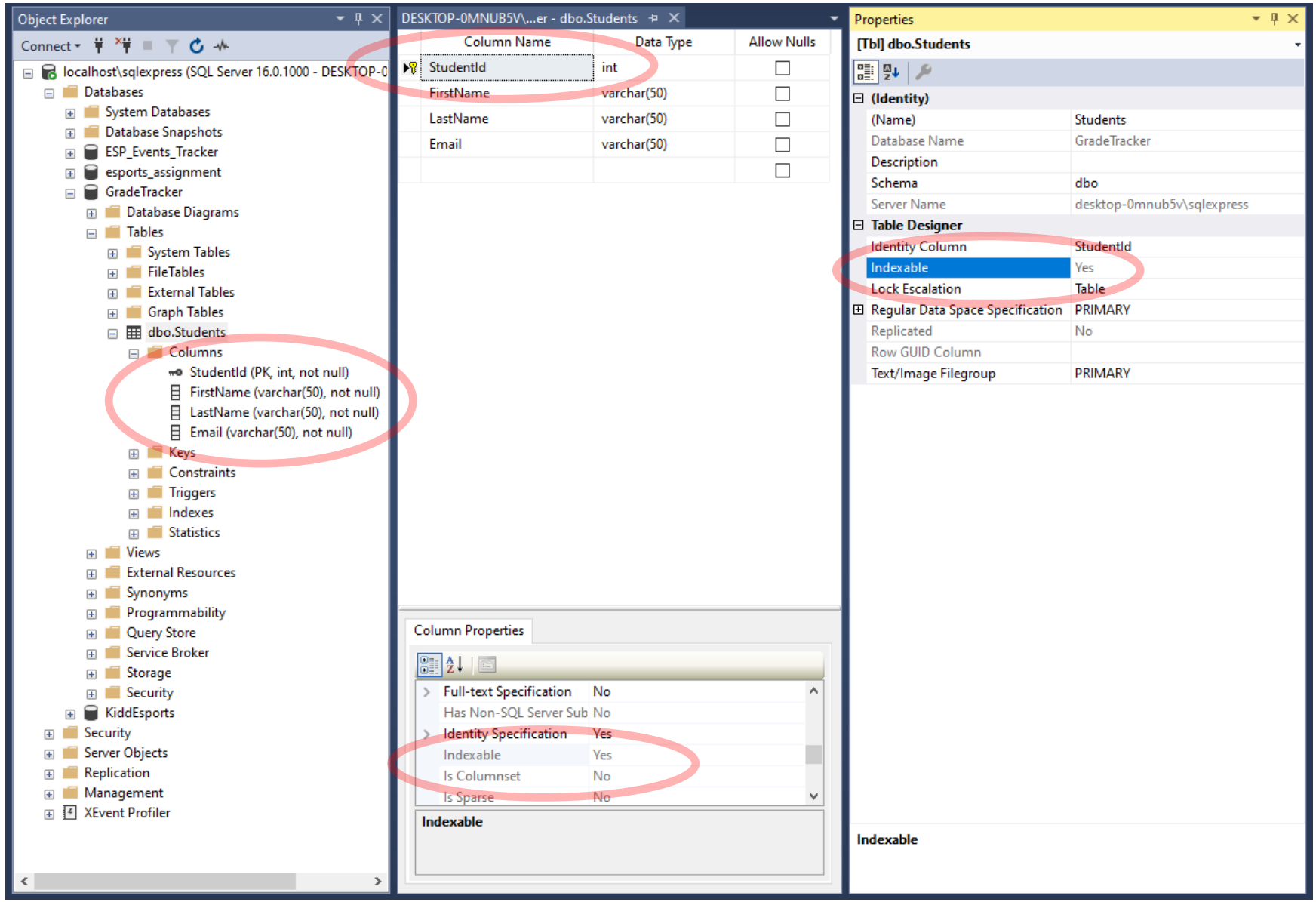
```
0 references
public class Student
{
    0 references
    public int StudentId { get; set; }
    0 references
    public string LastName { get; set; }
    0 references
    public string FirstName { get; set; }
    0 references
    public string Email { get; set; }
}
```

Attention!!! Do it BEFORE you save a table!!!

StudentId MUST be:

- **Primary Key**
- **Auto Incremental**





APP SETUP FOR SQL CONNECTIVITY

- Adding packages to project
  - Dapper
  - System.Data.SQLClient
  - System.Configuration.ConfigurationManager
- Creating app.config file
- Setting Up Connection String
- Build New project – DataManagement
- Setup Connection Between Classes
- Building Helper Class for Connection Setup
- Build DataManager Class

Expense tracker. Program structure

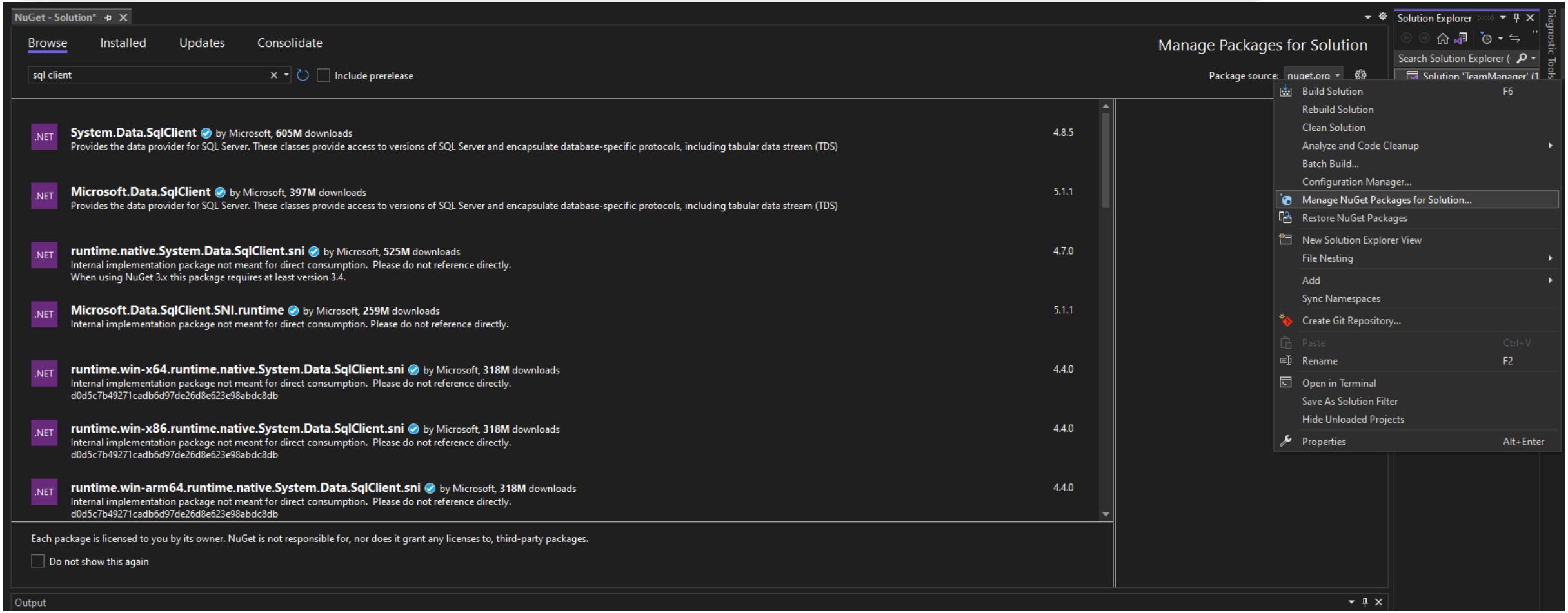
Modules

- **DataManagement project** – to create an open connection to our Server which can then be used by our application to send and receive data from our database
  - **Helper Class** - to be used to generate the SQL connections when we need them
  - **Data Model(s) class(es)** – representation(s) of the data from our table(s) to retrieve it and use it in our application
  - **DataManager class** – to CRUD

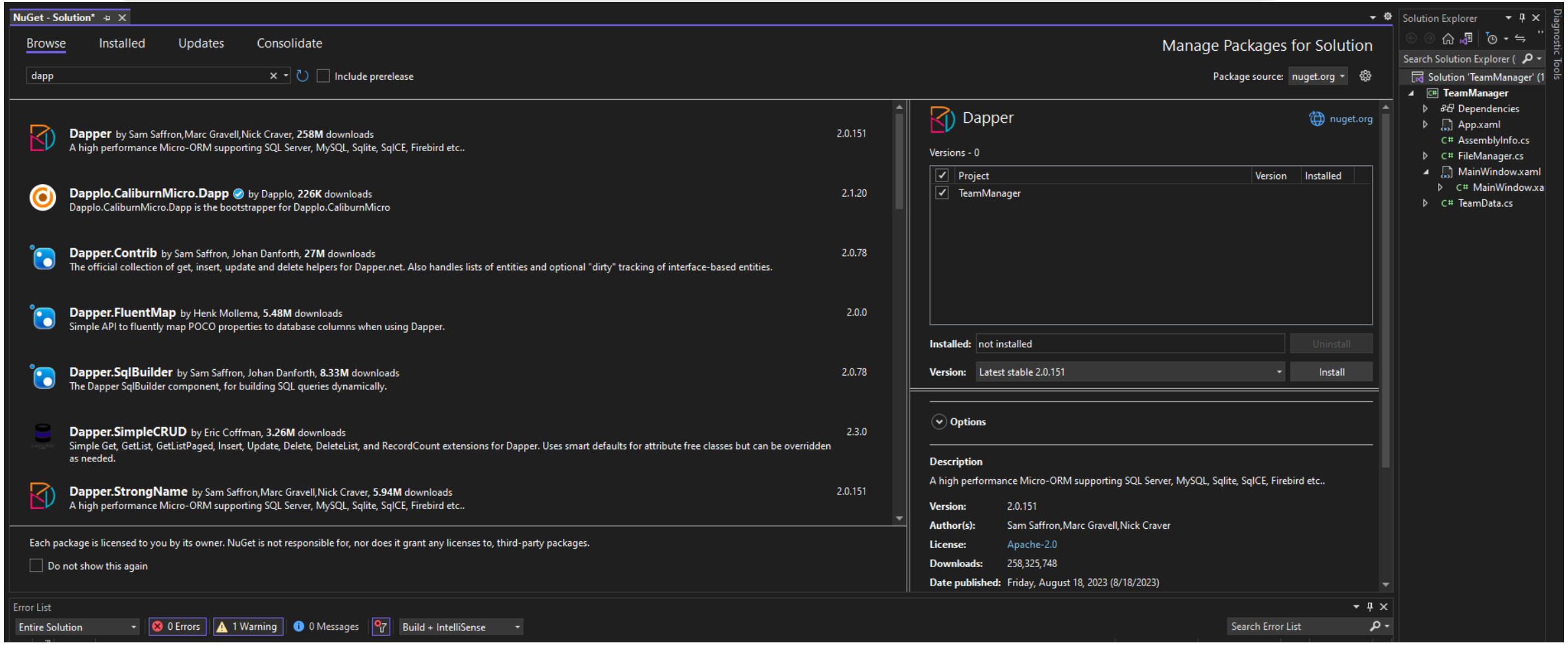
Packages

- Dapper
- System.Data.SQLClient
- System.Configuration.ConfigurationManager

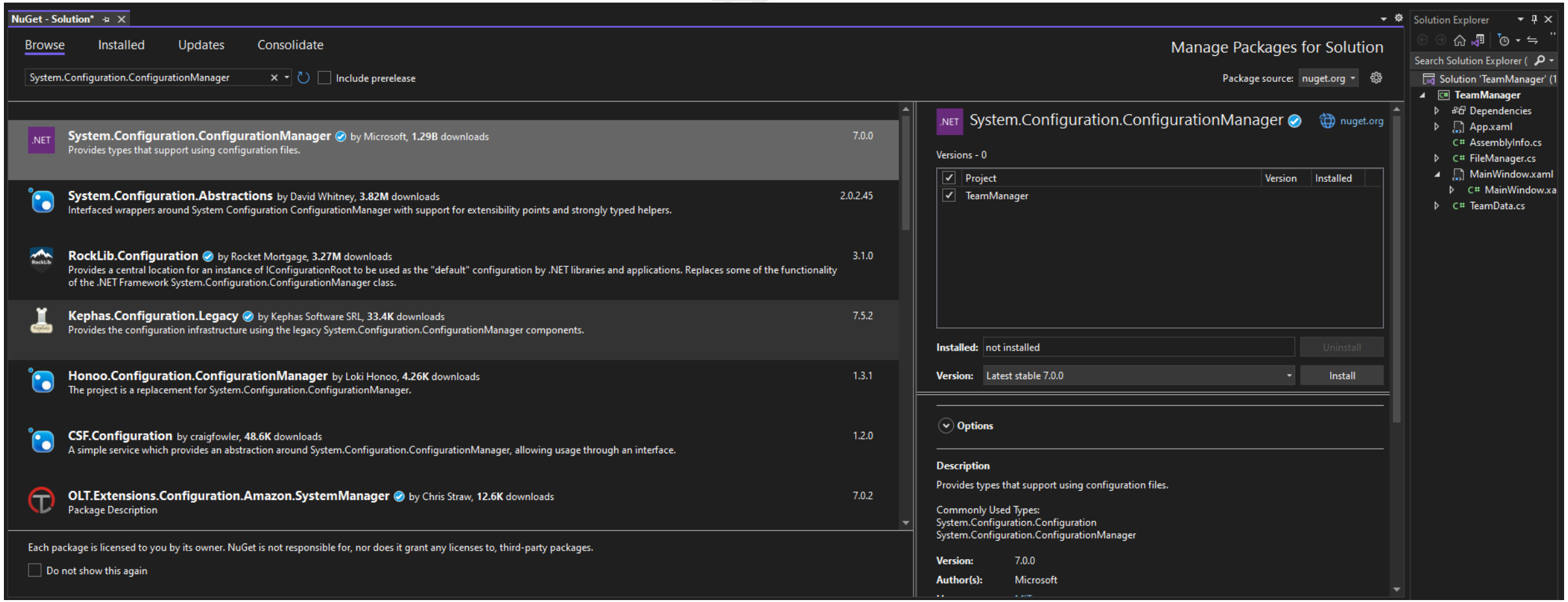
System.Data.SQLClient



Dapper


NuGet - Solution*
Browse
Installed
Updates
Consolidate
Manage Packages for Solution
dapp
Include prerelease
Package source: nuget.org
Dapper by Sam Saffron,Marc Gravell,Nick Craver, 258M downloads
A high performance Micro-ORM supporting SQL Server, MySQL, Sqlite, SqlCE, Firebird etc..
2.0.151
Dapplo.CaliburnMicro.Dapp by Dapplo, 226K downloads
Dapplo.CaliburnMicro.Dapp is the bootstrapper for Dapplo.CaliburnMicro
2.1.20
Dapper.Contrib by Sam Saffron, Johan Danforth, 27M downloads
The official collection of get, insert, update and delete helpers for Dapper.net. Also handles lists of entities and optional "dirty" tracking of interface-based entities.
2.0.78
Dapper.FluentMap by Henk Mollema, 5.48M downloads
Simple API to fluently map POCO properties to database columns when using Dapper.
2.0.0
Dapper.SqlBuilder by Sam Saffron, Johan Danforth, 8.33M downloads
The Dapper SqlBuilder component, for building SQL queries dynamically.
2.0.78
Dapper.SimpleCRUD by Eric Coffman, 3.26M downloads
Simple Get, GetList, GetListPaged, Insert, Update, Delete, DeleteList, and RecordCount extensions for Dapper. Uses smart defaults for attribute free classes but can be overridden as needed.
2.3.0
Dapper.StrongName by Sam Saffron,Marc Gravell,Nick Craver, 5.94M downloads
A high performance Micro-ORM supporting SQL Server, MySQL, Sqlite, SqlCE, Firebird etc..
2.0.151
Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.
Do not show this again
Dapper
nuget.org
Versions - 0
Project
Version
Installed
TeamManager
Installed: not installed
Uninstall
Version: Latest stable 2.0.151
Install
Options
Description
A high performance Micro-ORM supporting SQL Server, MySQL, Sqlite, SqlCE, Firebird etc..
Version: 2.0.151
Author(s): Sam Saffron,Marc Gravell,Nick Craver
License: Apache-2.0
Downloads: 258,325,748
Date published: Friday, August 18, 2023 (8/18/2023)
Solution Explorer
Search Solution Explorer
Solution 'TeamManager' (1
TeamManager
Dependencies
App.xaml
AssemblyInfo.cs
FileManager.cs
MainWindow.xaml
MainWindow.xa
TeamData.cs
Diagnostic Tools
Error List
Entire Solution
0 Errors
1 Warning
0 Messages
Build + IntelliSense
Search Error List

System.Configuration.ConfigurationManager



Creating app.config

- **app.config** is an XML file with many predefined configuration sections available and support for custom configuration sections. A "configuration section" is a snippet of XML with a schema meant to store some type of information.
- Settings can be configured using built-in configuration sections such as connectionStrings or appSettings. You can add your own custom configuration sections; this is an advanced topic, but very powerful for building strongly-typed configuration files.
- **Your project -> add -> item -> Application Configuration File**