Entity Framework vs World (June 2023)

Arturo Martínez (Yuxi Global – Associate Software Engineer)

Entity Framework vs World is one of the bounties of the Yggdrassil program, one of the objectives of this initiative is to create a knowledge base in which any professional could obtain information and ideas for different projects within the company.

The following document will describe with detail an approach with .Net framework as a principal core to work with millions of rows from an SQL query definition that contains all the fields of related tables from the AdventureWorks2019 SQL database.

I. Introduction

THIS document will describe the solution and the tasks developed to create a benchmark with a C# backend solution that allows knowing the time in milliseconds by selecting millions of records from an SQL database (AdventureWorks2019), with different strategies such as Linq2DB, Entity Framework7, Lazy Loading, Dapper Micro ORM, EF With AsNoTracking and Linq2SQL.

A C# console application was created inside a mapped virtual machine in which the AdventureWorks2019 SQL database was restored locally, the results of each scenario and the time in milliseconds are recorded in the Benchmark.Log table.

The following query has been defined to test the record query time from the database to our solution depending on the scenario chosen by the user:

```
Use the following Query (LongExecution Query cap it to a desired max result set ):
SELECT
 p.[Name],
 p.ProductNumber,
 th.*,
```

II. Development C# Backend Console Application / Database Context

A. Installed Libraries and NuGet packages.

Microsoft.EntityFrameworkCore.
Microsoft.EntityFrameworkCore.SqlServer.
Microsoft.EntityFrameworkCore.Tools
Microsoft.EntityFrameworkCore.Proxies
Dapper.

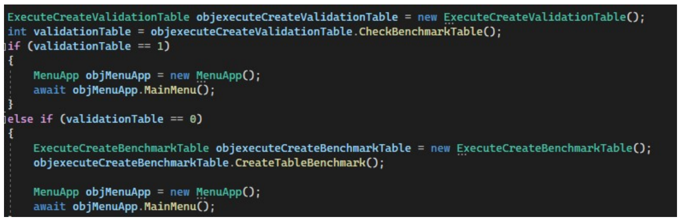
B. Database Modeling ant Context

Configured DATABASE FIRST to create classes, boxes and lines model that maps to AdventureWorks2019 Database (“AdventureWorks2019Context”).

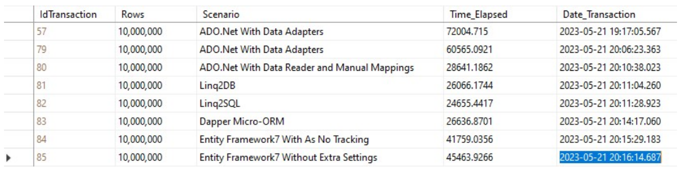
An Abstract class (“DataBaseHandler.cs”) was created as a Handler providing connection string to the Database and having count our OOP pillars an encapsulated class executes each scenario providing the requested information from provided query definition.

C. Table Creation on First Use:

On first application execution, an instruction will validate if the table (“Benckmark.Log”) exists on the database, if not, It will execute the DDL command to create the Benchmark table.



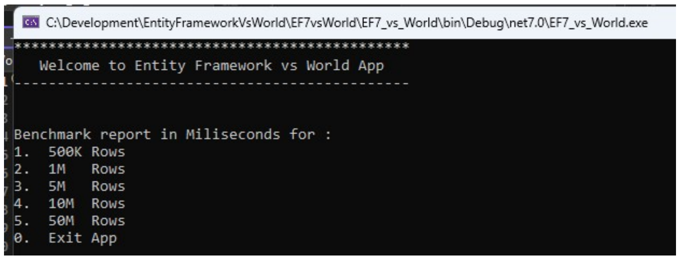
```
ExecuteCreateValidationTable objexecuteCreateValidationTable = new ExecuteCreateValidationTable();
int validationTable = objexecuteCreateValidationTable.CheckBenchmarkTable();
if (validationTable == 1)
{
    MenuApp objMenuApp = new MenuApp();
    await objMenuApp.MainMenu();
}
else if (validationTable == 0)
{
    ExecuteCreateBenchmarkTable objexecuteCreateBenchmarkTable = new ExecuteCreateBenchmarkTable();
    objexecuteCreateBenchmarkTable.CreateTableBenchmark();

    MenuApp objMenuApp = new MenuApp();
    await objMenuApp.MainMenu();
```



	IdTransaction	Rows	Scenario	Time_Elapsed	Date_Transaction
	57	10,000,000	ADO.Net With Data Adapters	72004.715	2023-05-21 19:17:05.567
	79	10,000,000	ADO.Net With Data Adapters	60565.0921	2023-05-21 20:06:23.363
	80	10,000,000	ADO.Net With Data Reader and Manual Mappings	28641.1862	2023-05-21 20:10:38.023
	81	10,000,000	Linq2DB	26066.1744	2023-05-21 20:11:04.260
	82	10,000,000	Linq2SQL	24655.4417	2023-05-21 20:11:28.923
	83	10,000,000	Dapper Micro-ORM	26636.8701	2023-05-21 20:14:17.060
	84	10,000,000	Entity Framework7 With As No Tracking	41759.0356	2023-05-21 20:15:29.183
▸	85	10,000,000	Entity Framework7 Without Extra Settings	45463.9266	2023-05-21 20:16:14.687

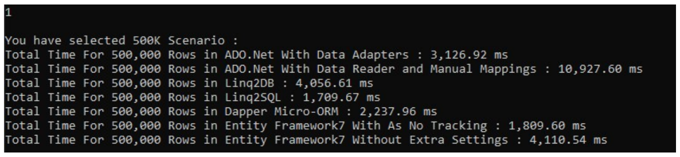
D. User Menu and Scenario Options

The user will be able to choose in the main menu the scenario that he wants to execute to know the time in milliseconds that each technology takes to bring the information from the database engine.



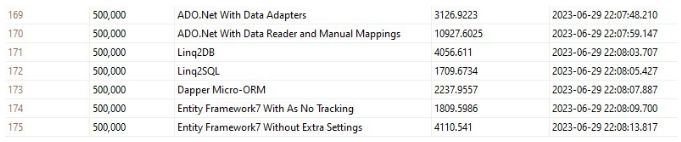
```
C:\Development\EntityFrameworkVsWorld\EF7vsWorld\EF7_vs_World\bin\Debug\net7.0\EF7_vs_World.exe
**************************************************
   Welcome to Entity Framework vs World App
--------------------------------------------------


Benchmark report in Miliseconds for :
1.  500K Rows
2.  1M   Rows
3.  5M   Rows
4.  10M  Rows
5.  50M  Rows
0.  Exit App
```

When the user has chosen his preferred scenario, the system will execute the query with different technologies and will show the result first as a Console Message.



```
1

You have selected 500K Scenario :
Total Time For 500,000 Rows in ADO.Net With Data Adapters : 3,126.92 ms
Total Time For 500,000 Rows in ADO.Net With Data Reader and Manual Mappings : 10,927.60 ms
Total Time For 500,000 Rows in Linq2DB : 4,056.61 ms
Total Time For 500,000 Rows in Linq2SQL : 1,709.67 ms
Total Time For 500,000 Rows in Dapper Micro-ORM : 2,237.96 ms
Total Time For 500,000 Rows in Entity Framework7 With As No Tracking : 1,809.60 ms
Total Time For 500,000 Rows in Entity Framework7 Without Extra Settings : 4,110.54 ms
```

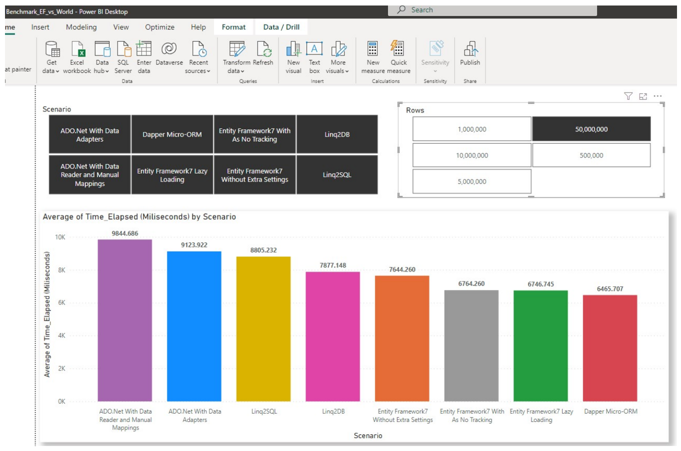
At the same time a module will Insert each transaction in Benchmark.Log table in SQL Server.



169	500,000	ADO.Net With Data Adapters	3126.9223	2023-06-29 22:07:48.210
170	500,000	ADO.Net With Data Reader and Manual Mappings	10927.6025	2023-06-29 22:07:59.147
171	500,000	Linq2DB	4056.611	2023-06-29 22:08:03.707
172	500,000	Linq2SQL	1709.6734	2023-06-29 22:08:05.427
173	500,000	Dapper Micro-ORM	2237.9557	2023-06-29 22:08:07.887
174	500,000	Entity Framework7 With As No Tracking	1809.5986	2023-06-29 22:08:09.700
175	500,000	Entity Framework7 Without Extra Settings	4110.541	2023-06-29 22:08:13.817

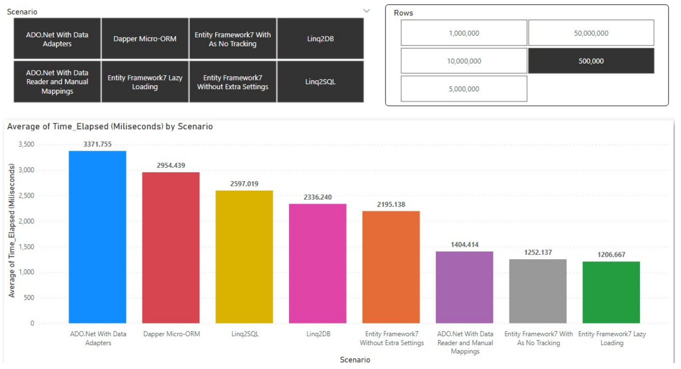
E. *Benchmark Reporting*

Microsoft Power BI has been configured with the Benchmark.Log table as Data Source, more than 50 attempts by each scenario, reported average from each configuration and this is shown as a bar visual in Microsoft Power BI (Report Included in the solution.)

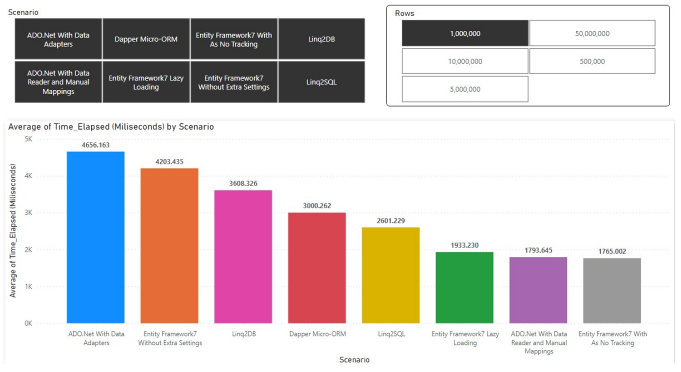


III. Benchmark Results By Scenario

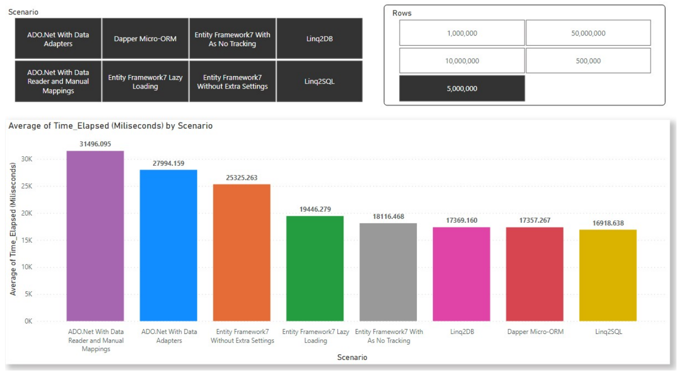
500,000 Rows:



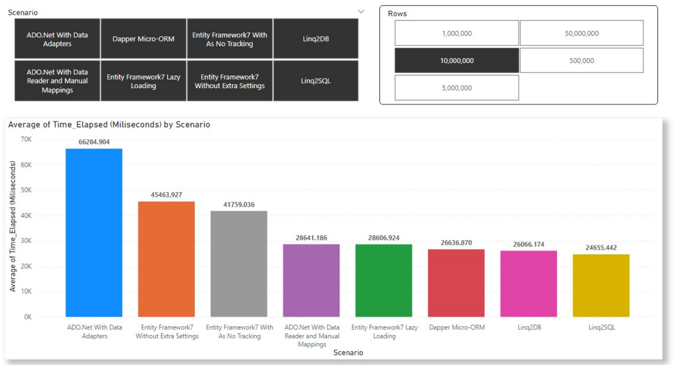
1,000,000 Rows:



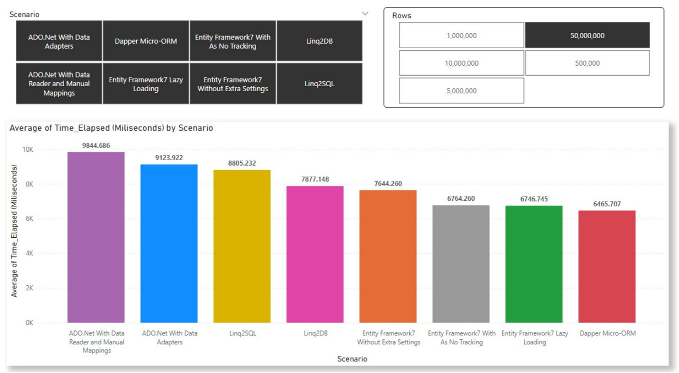
5,000,000 Rows:



10,000,000 Rows:



50,000,000 Rows:



IV. CONCLUSIONS

After executing more than 50 attempts by each scenario the solution will provide a detailed average of milliseconds executed on each technology and number of rows chosen by the user.

Dapper Micro ORM and Entity Framework with Lazy loading features and Linq raw connections are the best scenarios when the solution must work with more than 5 million of rows. They have similar execution times, and all the workload is pointing to the database engine.

Working with 500,000 rows or less than 5 million, Raw connections as ADO.Net take a high importance with performance and time elapsed.

Many references from developer's forums, stack overflow suggestions and Microsoft documents suggest identifying the best feature and try to use all the capabilities from Entity Framework 7 and all its benefits to show a very high volume of information. It is important to validate and have a simulation of the possible volume of information to choose the best scenario for the client or project.

V. RESOURCES AND REPOSITORIES

A. *GitHub Repository*

Solution Public Repository: https://github.com/EdgarArturoMartinez/EF7vsWorld

B. *Reporting PBI File*

Reporting PBI File: https://github.com/EdgarArturoMartinez/EF7vsWorld/tree/master/EF7_vs_World/Resources

C. *Documentation*

Solution Document: https://github.com/EdgarArturoMartinez/EF7vsWorld/tree/master/EF7_vs_World/Resources

REFERENCES

[1] Working with Millions of rows – Pedro de Leon YouTube Channel https://www.youtube.com/watch?v=RXNEjDEwzHI

[2] Lazy Loading in Entity Framework. - https://www.entityframeworktutorial.net/lazyloading-in-entity-framework.aspx

[3] Using the elastic database client library with Dapper. - https://learn.microsoft.com/en-us/azure/azure-sql/database/elastic-scale-working-with-dapper?view=azuresql

[4] Tracking vs, No-Tracking Queries. - https://learn.microsoft.com/en-us/ef/core/querying/tracking

[5] LINQ to SQL tools. - https://learn.microsoft.com/en-us/visualstudio/data-tools/linq-to-sql-tools-in-visual-studio2?view=vs-2022

[6] Entity Data Model. - https://learn.microsoft.com/en-us/dotnet/framework/data/adonet/entity-data-model

[7] Using ORM - https://www.learnentityframeworkcore.com/

[8] Maximizing C# Database performance with Dapper. - https://medium.com/c-sharp-progarmming/maximizing-c-database-performance-with-dapper-7a4562d3a47a