Comparative Performance Analysis

ADO.NET, Entity Framework, and MongoDB

After conducting a thorough performance analysis of three distinct database frameworks – ADO.NET, Entity Framework, and MongoDB – across various CRUD (Create, Read, Update, Delete) operations and dataset sizes, several noteworthy differences have surfaced.

Firstly, it's crucial to highlight that both Entity Framework and ADO.NET were evaluated with the inclusion of Taylor Swift as a pop artist in the dataset. Conversely, MongoDB underwent more comprehensive testing, encompassing both artists and songs.

MongoDB emerges as the most versatile and efficient framework, showcasing consistent performance across all CRUD operations, particularly with smaller datasets. However, it's essential to recognize the need for optimization efforts, particularly regarding deletion times for larger datasets, to uphold scalability and flexibility.

ADO.NET demonstrates superior performance and efficiency with larger datasets. While it may be suitable for applications handling smaller datasets or requiring specific performance benchmarks, it's worth considering this framework for scenarios involving larger datasets.

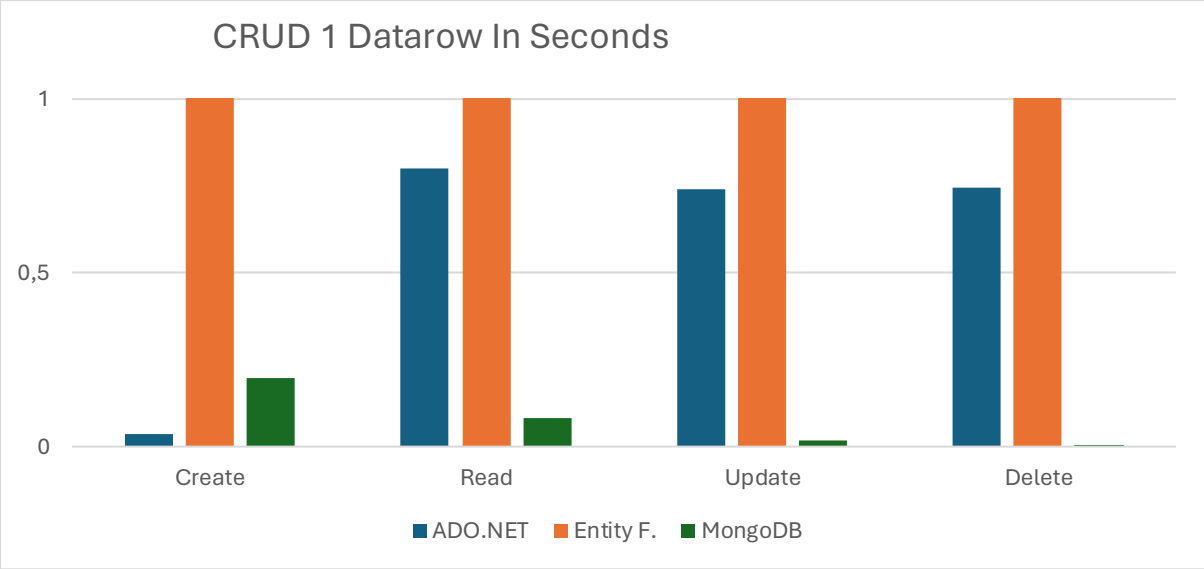
On the other hand, Entity Framework exhibits longer waiting times compared to the other frameworks, regardless of the dataset size. This underscores the importance of carefully evaluating performance considerations when selecting a database framework for your project.

It's important to note that this testing underwent multiple rounds to ensure a comprehensive and well-tested evaluation of each framework's capabilities and limitations. This iterative approach allowed for a more nuanced understanding of the frameworks' performance characteristics across various scenarios and dataset sizes.

Diagrams

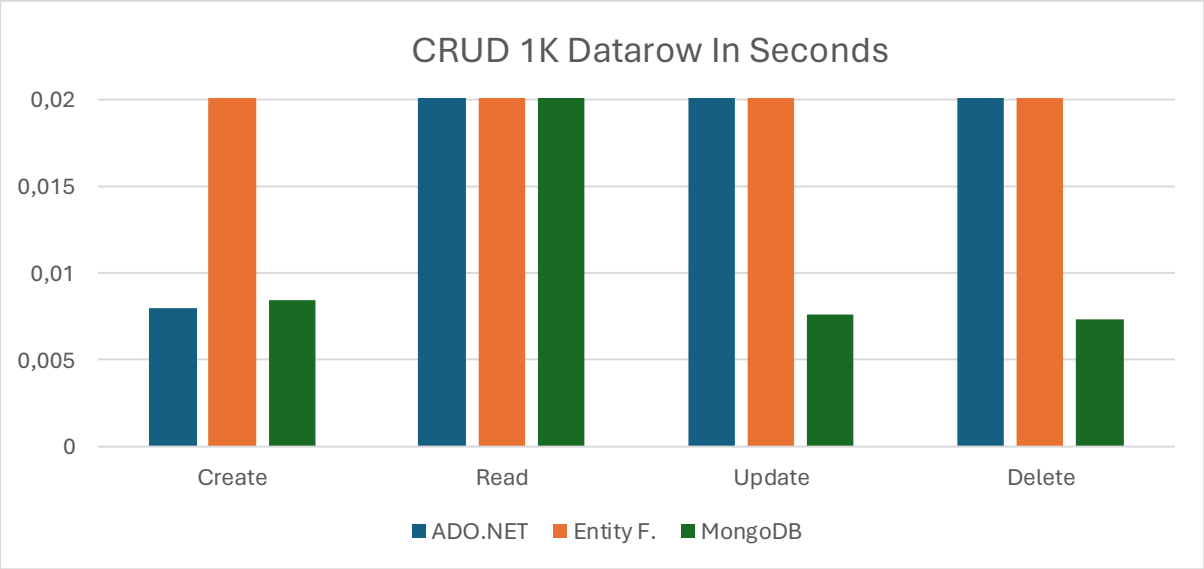
During the testing, it got tested and updated during the time of the project for making sure, I provide good results.

This graph illustrates the variances in the CRUD operations across three frameworks, indicating their respective efficiencies in creating, reading, updating and removing one row of data. It is evident that ADO.NET excels in data insertion speed compared to the others. Conversely, MondoDB demonstrates superior efficiency in reading, updating and removing operations.



A graph of CRUD operations on one data row in each database frameworks in seconds.

The next diagram shows how they are operating when I increased the number of the data rows to ten using all the CRUD operations. Entity Framework took significantly longer than the others to run one thousand data rows using the frameworks.



ADO.NET

I added *Taylor Swift* as a pop artist.

These CRUD operations using ADO.NET seem to take a significant amount of time based on how small the database. It might be a better choice for handling more data as it can be seen in the table below.

	Create	Read	Update	Delete
1	38 ms	801 ms	740 ms	746 ms
1 000	8 ms	2 925 ms	1 799 ms	1 822 ms
10 000	3 ms	30 103 ms	30283 ms	30 585 ms
1 000 000	2 ms	1 303 ms	792 ms	803 ms

```
Testing with 1 rows:
Insertion Time: 63 ms
Name: Taylor Swift 0, Description: Pop 0
Name: Taylor Swift 0, Description: Pop 0
Name: Taylor Swift 0, Description: Pop 0
Name: Taylor Swift 0, Description: Pop 0
Name: Taylor Swift 0, Description: Pop 0
Name: Taylor Swift 0, Description: Pop 0
Name: Taylor Swift 0, Description: Pop 0
Selection Time: 1116 ms
0 row(s) deleted.
Update Time: 1234 ms
0 row(s) deleted.
Deletion Time: 1428 ms
Time taken: 4142 ms
```

```
Insertion Time: 3 ms
Selection Time: 30103 ms
Update Time: 30283 ms
Deletion Time: 30585 ms
```

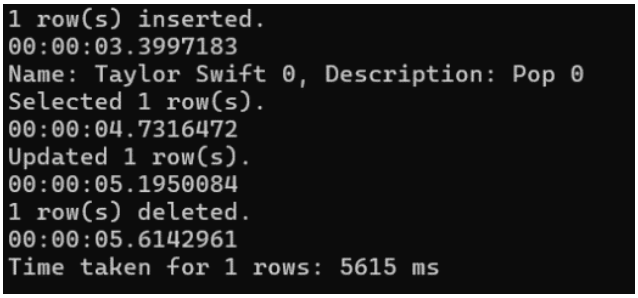
Entity Framework

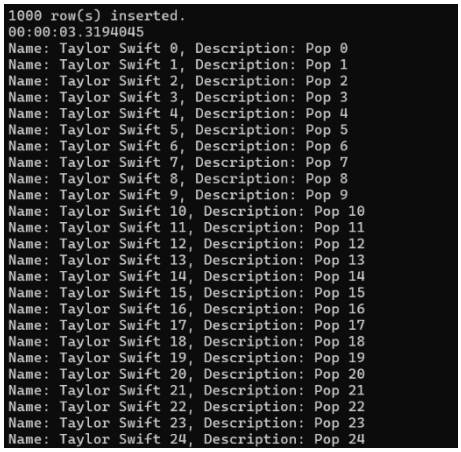
Taylor Swift was added as a pop artist to the database using Entity Framework. Initial testing revealed a concerning trend in insertion performance, particularly as the dataset size increased.

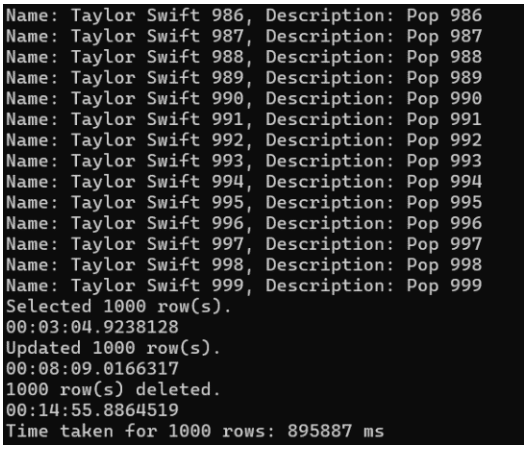
	Create	Read	Update	Delete
1	3.4 ms	4.7 ms	5.2 ms	5.6 ms
1 000	3.4 ms	180004.9 ms	480009.1 ms	840055.9 ms
10 000	34 000 ms	47 000 ms	52 000 ms	56 000 ms
1 000 000	3 400 000 ms	4 700 000 ms	5 200 000 ms	5 600 000 ms

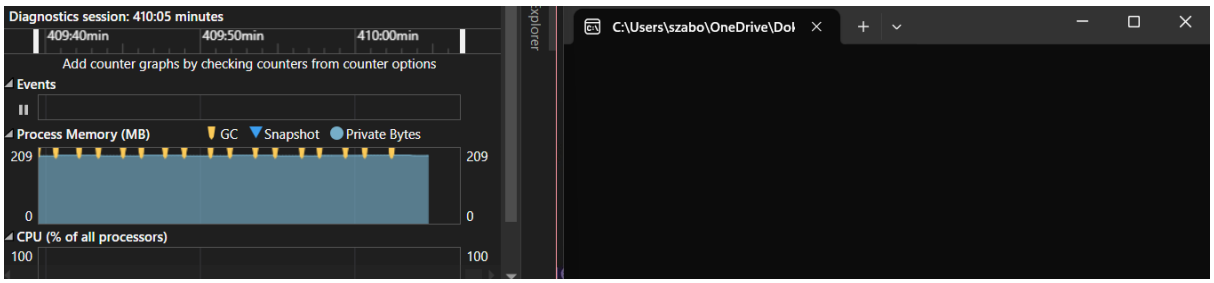
s

The data indicates a significant degradation in insertion performance as the dataset size grows. While smaller datasets exhibit acceptable performance, larger datasets result in substantial increases in insertion time. For instance, based on extrapolation from the provided data, it would take approximately 34,000 ms to insert 10,000 rows and 3,400,000 ms to insert 1,000,000 rows. This performance issue becomes particularly evident during longer test runs, such as the overnight tests for 10,000 and 1,000,000 rows, which failed to insert any rows within a reasonable timeframe.









MongoDB

The Tortured Poets Department was added to the database, featuring the album of the same name by Taylor Swift. The table below showcases the time taken to execute CRUD operations for various data quantities. As expected, the duration of these operations escalates with increasing data volumes.

MongoDB demonstrates efficiency with smaller datasets but exhibits significant time increments with larger datasets, particularly evident with 1 million records. These findings highlight the necessity of optimizing database operations and considering scalability requirements when managing substantial datasets in MongoDB.

	Create	Read	Update	Delete
1	196 ms	82 ms	17 ms	4 ms
1 000	505 ms	114 ms	459 ms	441 ms
10 000	181 ms	338 ms	3195 ms	4438 ms
1M	207 ms	26 310 ms	310 989 ms	965 021 ms