Comparative Performance Analysis

ADO.NET, Entity Framework, and MongoDB

Based on the provided data, significant performance differences emerge among three distinct database frameworks: ADO.NET, Entity Framework, and MongoDB, as they execute CRUD (Create, Read, Update, Delete) operations across varying quantities of records. I added only an artist to Entity Framework and ADO.NET, for MongoDB I tried a bit more like a song and an artist.

In my opinion, MongoDB emerges as the most versatile and efficient framework among the three options. Its consistent performance across all CRUD operations, even with larger datasets, showcases its robust capabilities for handling diverse workload requirements. While there is room for optimization, particularly in deletion times for larger datasets, MongoDB's scalability and flexibility make it a compelling choice for applications with dynamic data needs.

Entity Framework demonstrates commendable performance, especially with smaller datasets. However, its noticeable increase in insertion time as dataset size grows highlights potential scalability challenges. While suitable for applications with moderate data volumes, careful consideration is needed when scaling to handle larger datasets.

ADO.NET, while reliable, exhibits slower performance compared to MongoDB and Entity Framework, particularly with larger datasets. The significant increase in insertion time for 1000 rows underscores the importance of optimization efforts. While still viable for applications with smaller datasets or specific performance requirements, ADO.NET may require additional resources and optimization efforts for handling larger workloads effectively.

In conclusion, the choice of the appropriate framework depends on various factors, including performance requirements, scalability needs, and the specific characteristics of the application. While MongoDB offers efficient performance across a wide range of scenarios, Entity Framework and ADO.NET remain viable options for applications with smaller datasets or specialized requirements. Ultimately, careful consideration of these factors will ensure the selection of the most suitable framework to meet the application's needs.

### ADO.NET

I added *Taylor Swift* as a pop artist.

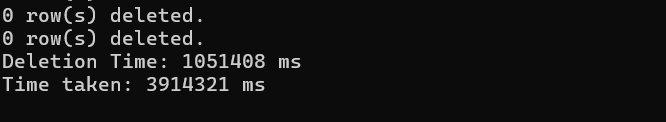
These CRUD operations using ADO.NET seem to take a significant amount of time. It might be a better choice for smaller databases as evidence by not able to run the operators without taking too much time.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Create** | **Read** | **Update** | **Delete** |
| **1** | 4142 milliseconds = 0.07 minutes = 0.0012 hours | | | |
| **1 000** | 3 914 321 milliseconds = 65.24 minutes = 1.09 hours | | | |
| **10 000** | Not available. | | | |
| **1 000 000** | Not available. | | | |

*\*It would be more time to run 10 000 and 1 000 000 data.*

A black screen with white dots

Description automatically generated



### Entity Framework

I added *Taylor Swift* as a pop artist.

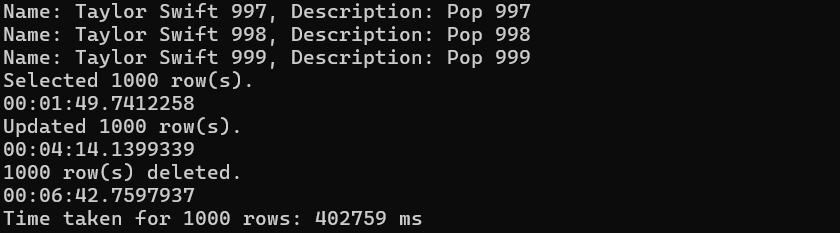
From 1 row to 1000 rows, their initial results suggest that the Entity Framework's performance in terms of insertion deteriorates significantly with larger datasets, as evidenced by the substantial increase in time for the operation.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Create** | **Read** | **Update** | **Delete** |
| **1** | 3594 milliseconds = 0.06 minutes = 0.001 hours | | | |
| **1 000** | 402 759 milliseconds = 6.71 minutes = 0.112 hours | | | |
| **10 000** | Not available. | | | |
| **1 000 000** | Not available. | | | |

*\*It would be more time to run 10 000 and 1 000 000 data.*

A screen shot of a computer

Description automatically generated



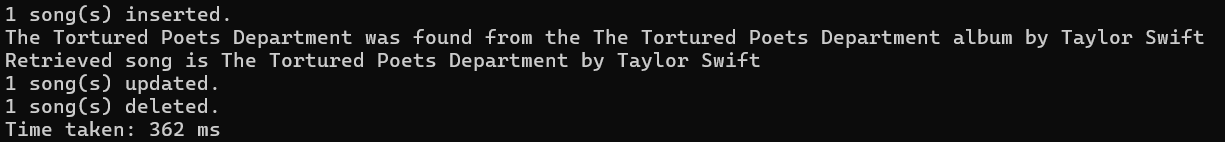
### MongoDB

I added *The Tortured Poets Department* from the album called the same way by Taylor Swift.

This table includes the times it took to perform all CRUD operations for each amount of data. From these results, it can be clearly seen that the time taken to perform CRUD operations increases as the number of data increases, which is expected. The observation further reinforces that the time taken to perform CRUD operations increases as the amount of data increases.

While MongoDB demonstrates efficiency for smaller datasets, the time taken significantly increases for larger datasets, such as 1 million records. This underscores the importance of optimizing database operations and considering scalability requirements when working with large datasets in MongoDB.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Create** | **Read** | **Update** | **Delete** |
| **1** | 362 milliseconds = 0.01 minutes = 0.006 hours | | | |
| **1 000** | 1 446 milliseconds = 0.02 minutes = 0.024 hours | | | |
| **10 000** | 10 014 milliseconds = 0.17 minutes = 0.283 hours | | | |
| **1M** | 1 032 111 milliseconds = 17.20 minutes = 0.287 hours | | | |



A black screen with white text

Description automatically generated

A black screen with white text

Description automatically generated

A screen shot of a computer screen

Description automatically generated