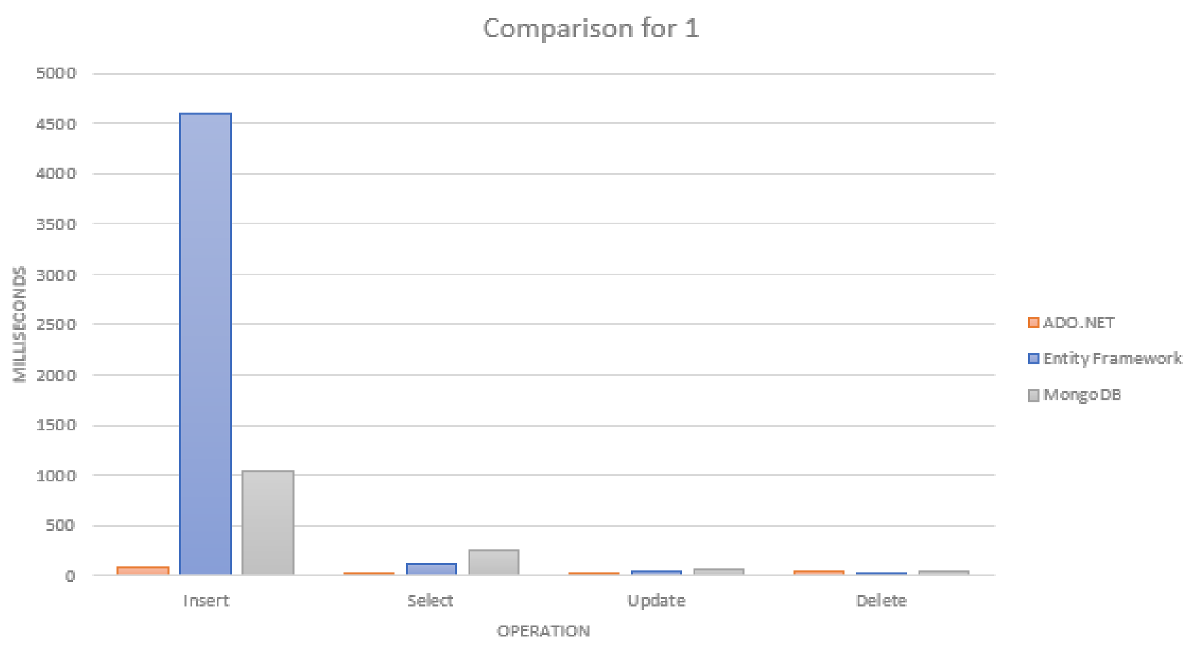
Data gathered from the investigation.

ADO.NET		
Operation	Amount	Result(milliseconds)
Insert	1	76
Select	1	3
Update	1	8
Delete	1	38
Insert	100	1980
Select	100	520
Update	100	2139
Delete	100	1042
Insert	100000	57444
Select	100000	19986
Update	100000	56437
Delete	100000	113096
Insert	500000	217986
Select	500000	67999
Update	500000	158787
Delete	500000	286609

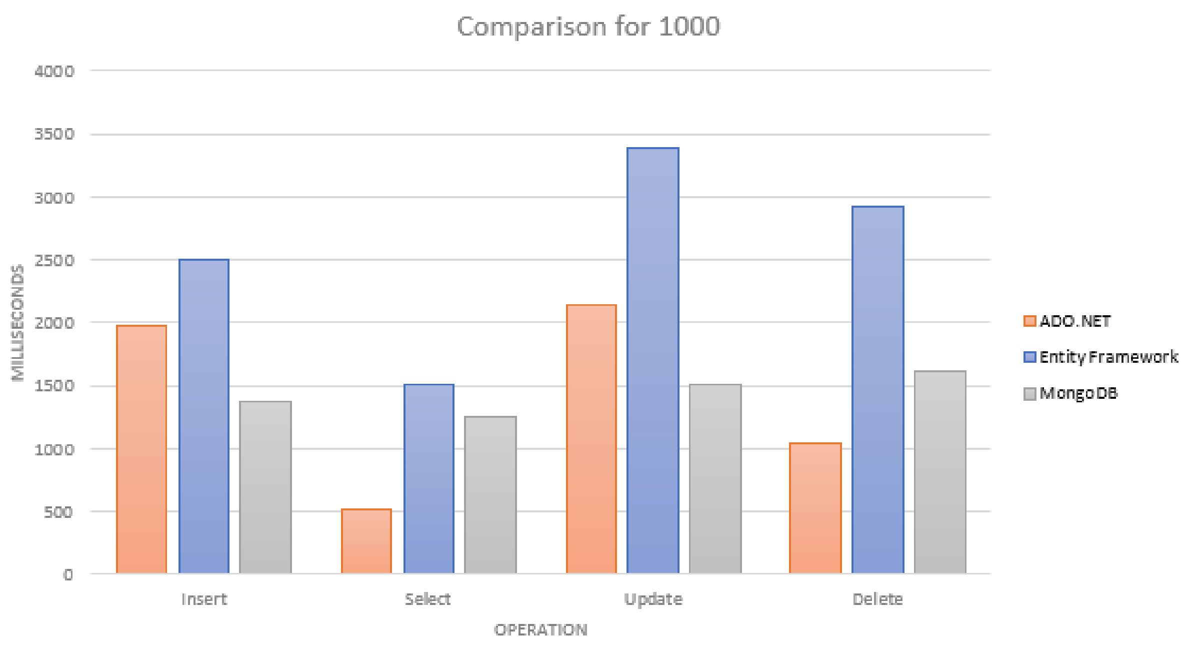
Entity Framework		
Operation	Amount	Result(milliseconds)
Insert	1	4597
Select	1	119
Update	1	36
Delete	1	15
Insert	100	2506
Select	100	1502
Update	100	3384
Delete	100	2921
Insert	100000	388641
Select	100000	198873
Update	100000	526342
Delete	100000	477521
Insert	500000	1613848
Select	500000	983395
Update	500000	2801454
Delete	500000	2542509

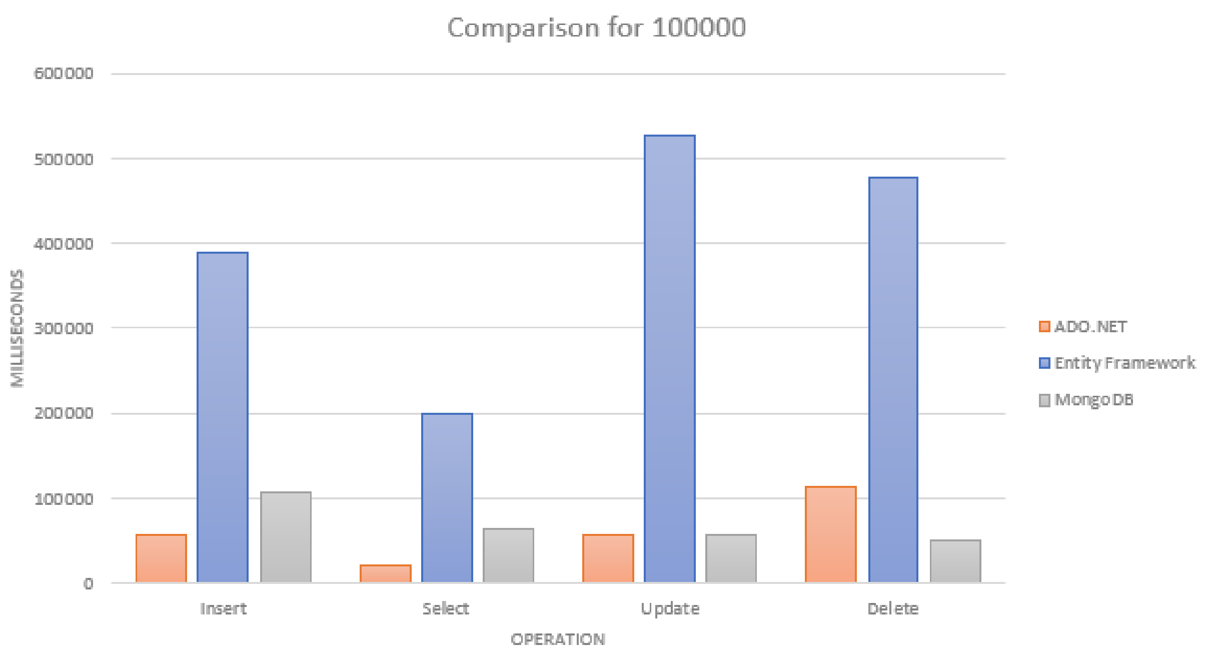
MongoDB		
Operation	Amount	Result(milliseconds)
Insert	1	1041
Select	1	247
Update	1	60
Delete	1	35
Insert	100	1369
Select	100	1249
Update	100	1508
Delete	100	1613
Insert	100000	105866
Select	100000	63633
Update	100000	55924
Delete	100000	51121
Insert	500000	237144
Select	500000	204098
Update	500000	186803
Delete	500000	215646

Graphs



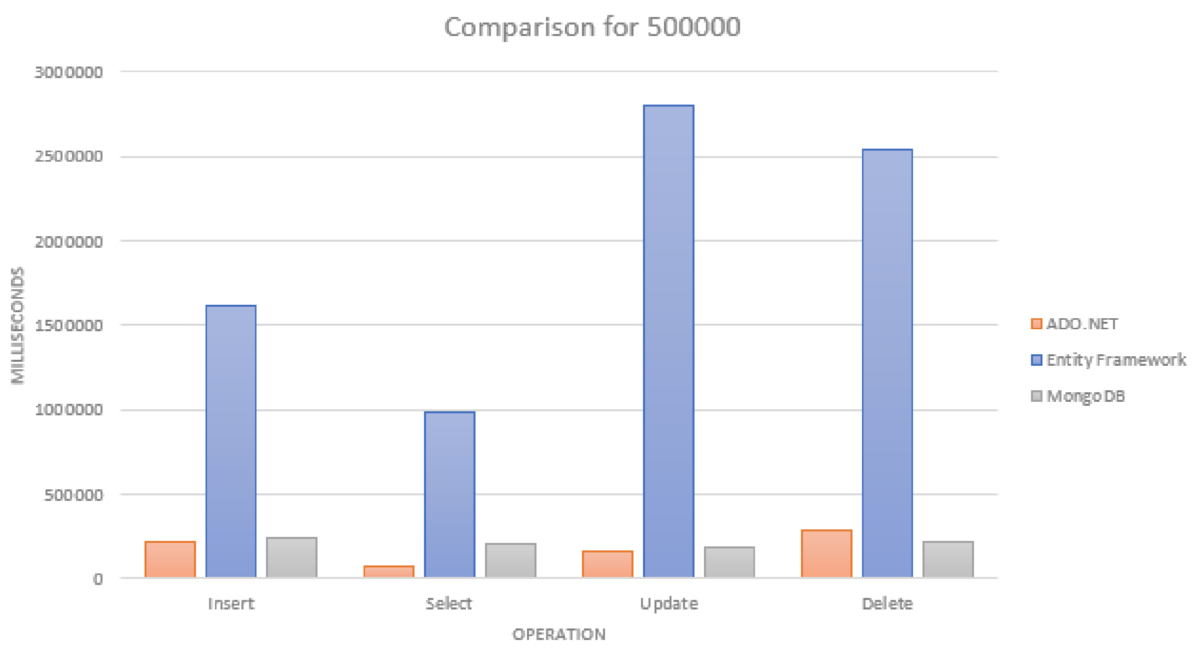
Entity frameworks first run takes 4.5 times as long as MongoDB but after that it stabilizes. With some research I found that this is due to EF maintaining metadata from the models(Tables) created upon runtime.



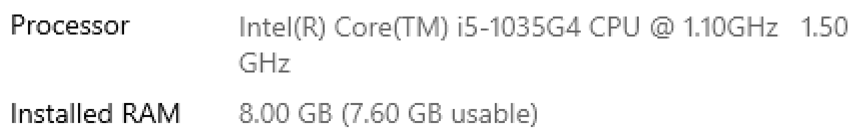


Entity Framework is as extremely slow when it comes to inserting large amounts (100000 and 500000) because of the SaveChanges method.

ADO.NET is faster than MongoDB when it comes to insert, select but tied with update and MongoDB takes the lead with delete this is because of Indexes



Specifications



Processor	Intel(R) Core(TM) i5-1035G4 CPU @ 1.10GHz 1.50 GHz
Installed RAM	8.00 GB (7.60 GB usable)

Steps taken to ensure data is reliable.

During the investigation teams and Firefox were the only extra applications running, and the number of tabs open was consistent.