CS315 Assignment

Aman Tayal
180074

---

1) Queries

SQL

A) SELECT * FROM A WHERE A1 <= 50;

B) SELECT * FROM B ORDER BY B3;

C) SELECT B2,COUNT(*) FROM B GROUP BY B2;

D) SELECT B1,B2,B3,A2 FROM B INNER JOIN A ON B.B2 = A.A1;

MongoDB

A) db.A.find({ A1: {$lte : 50}})

B) db.B.aggregate([{ $sort : {B3:1} }])

C) db.B.aggregate([{ $group: {_id:"$B2", count: {$sum: 1}} }])

D) db.B.aggregate(
```
    [
      { $lookup: { from: "A", localField: "B2", foreignField: "A1", as: "_A" }},
      { $project: { B1:1, B2:1,B3:1,"_A.A2":1 }}
    ] ,
    {"allowDiskUse":true}
  )
```

(allowDiskUse is true as required for large dataset in MongoDB)

2) Time Taken by each query for each database

Databases for roll no. 180074, a=0, b = 7, c=4
9 numbers → {0,0,0,0.4,3,0,3,1}
Databases →

1. A-100.csv, B-100-3-0.csv
2. A-100.csv, B-100-5-0.csv
3. A-100.csv, B-100-10-0.csv
4. A-1000.csv, B-1000-5-0.csv
5. A-1000.csv, B-1000-10-4.csv
6. A-1000.csv, B-1000-50-3.csv
7. A-10000.csv, B-10000-5-0.csv
8. A-10000.csv, B-10000-50-3.csv
9. A-10000.csv, B-10000-500-1.csv

For each databases we run each query on all engines 7 times except for 9th database which is only run 5 times on mongoDB for query D. We calculated mean and standard deviation values after removing outliers.

Table

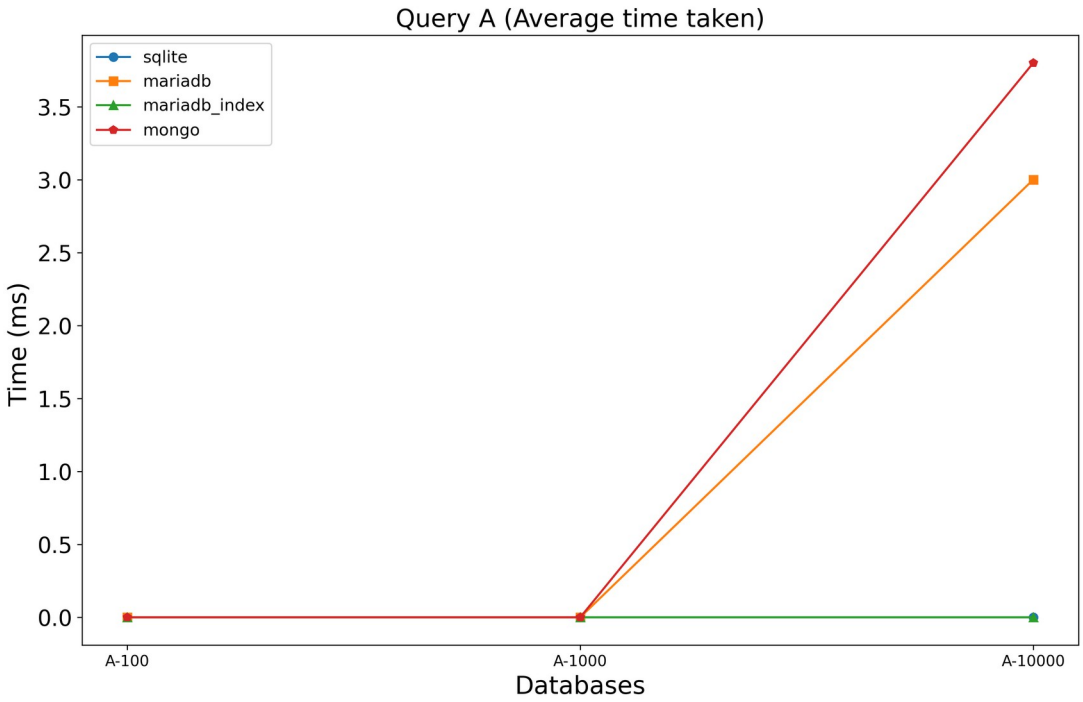
In table all mean and standard deviation is reported in milli seconds
column corresponds to each of 9 databases mention above

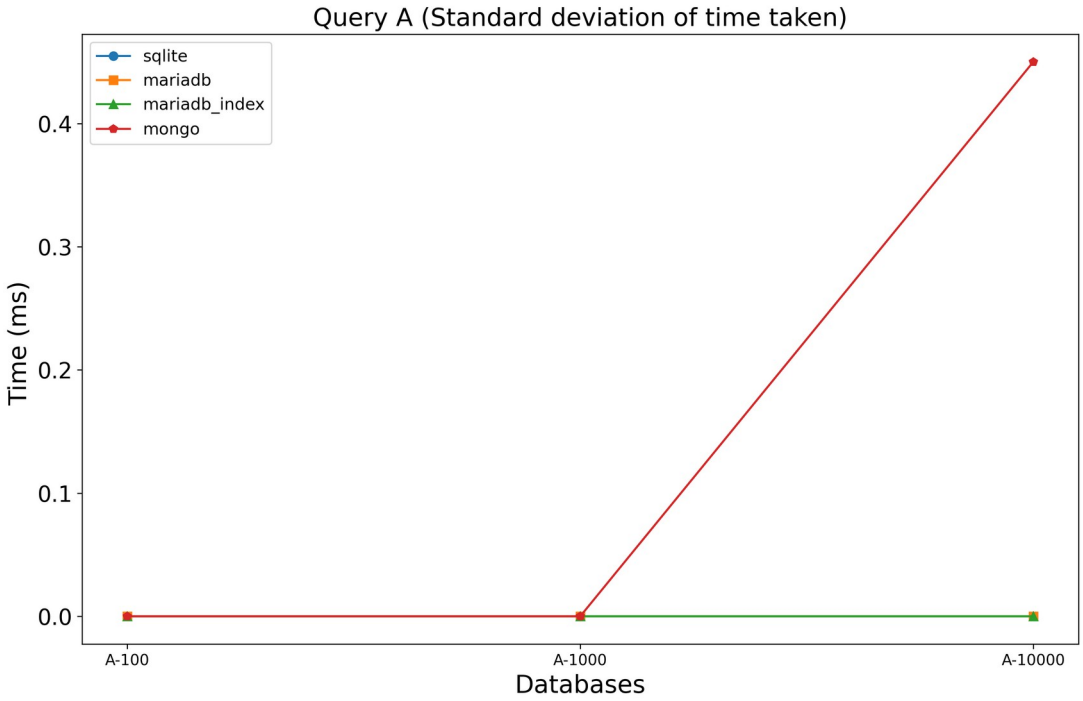
		B-100-3-0		B-100-5-0		B-100-10-0		B-1000-5-0		B-1000-10-4		B-1000-50-3		B-10000-5-0		B-10000-50-3		B-10000-500-1	
		mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
queryA	sqlite	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	mariadb	0.2	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0	3.0	0.0	3.0	0.0
	mariadb_index	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	mongo	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.4	0.6	3.4	0.6	3.8	0.5
queryB	sqlite	0.2	0.5	0.2	0.5	0.6	0.6	3.0	0.0	5.6	0.6	25.4	0.6	34.8	0.5	313.6	2.1	3251.8	51.9
	mariadb	0.0	0.0	0.0	0.0	1.0	0.0	4.4	0.6	7.0	0.0	37.0	0.0	46.0	1.0	406.8	0.8	4779.2	28.1
	mariadb_index	0.0	0.0	0.0	0.0	0.0	0.0	1.4	0.6	2.4	0.6	10.0	0.0	13.2	0.5	87.2	0.8	1322.6	245.1
	mongo	0.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0	6.0	0.0	32.8	0.8	43.6	0.6	421.8	5.3	579.0	5.3
queryC	sqlite	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.6	0.6	4.6	0.6	10.4	0.6	48.4	1.7	550.0	4.6
	mariadb	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	2.8	0.5	11.0	0.0	18.4	0.9	105.0	0.0	1130.8	1.3
	mariadb_index	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	4.0	0.0	7.0	0.0	41.0	0.0	698.6	24.4
	mongo	0.0	0.0	0.0	0.0	0.0	0.0	4.4	0.6	7.8	0.5	32.6	0.9	55.6	1.5	306.8	1.6	3314.4	89.7
queryD	sqlite	0.7	0.6	0.3	0.6	0.0	0.0	2.6	0.6	4.4	0.6	19.8	0.5	26.0	0.0	189.2	2.6	1932.8	14.5
	mariadb	1.8	0.5	2.0	0.0	4.0	0.0	179.8	0.5	381.0	0.7	1627.4	108.8	17006.4	4.6	125365.8	12.8	1237680.0	4901.8
	mariadb_index	0.0	0.0	0.4	0.6	1.0	0.0	6.0	0.0	9.2	0.5	39.6	1.3	57.8	0.5	421.6	7.4	7868.2	1450.2
	mongo	21.4	0.9	28.0	0.0	50.4	0.9	1569.8	1.3	2776.0	2.5	12198.8	23.0	144239.4	316.9	1067537.8	1896.8	10975498.0	413809.6

3) Graphs

1) For query A, time taken by each engine on each database

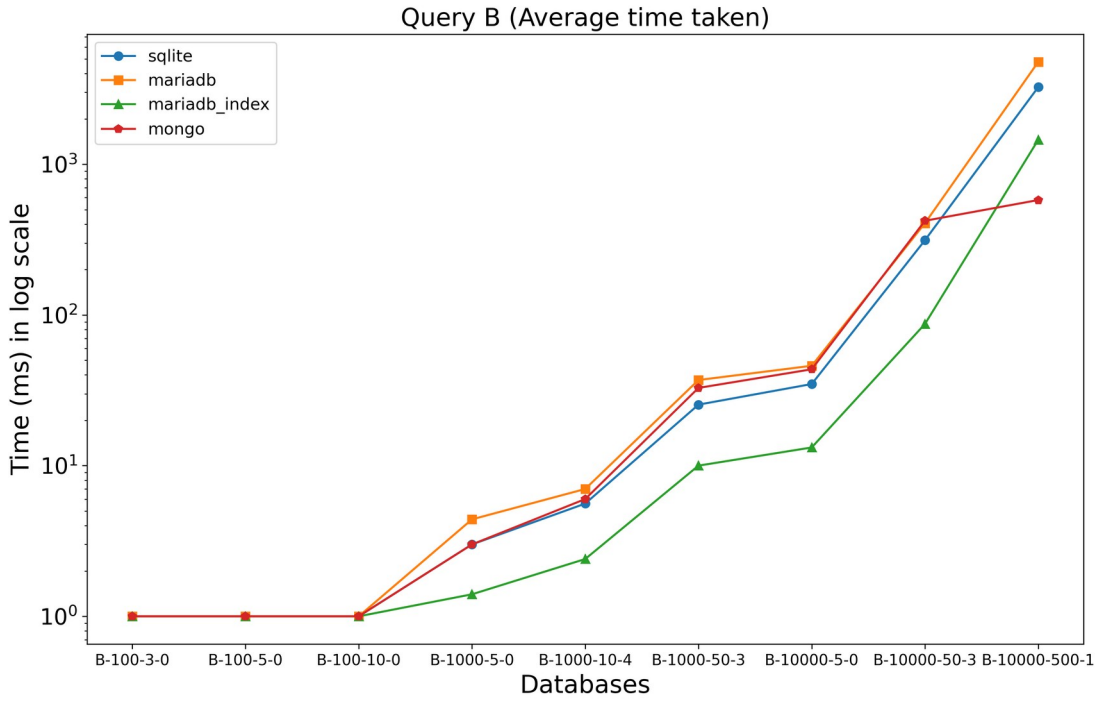
as queryA only depends on table A so we only take A-100,A-1000,A-10000 on x axis

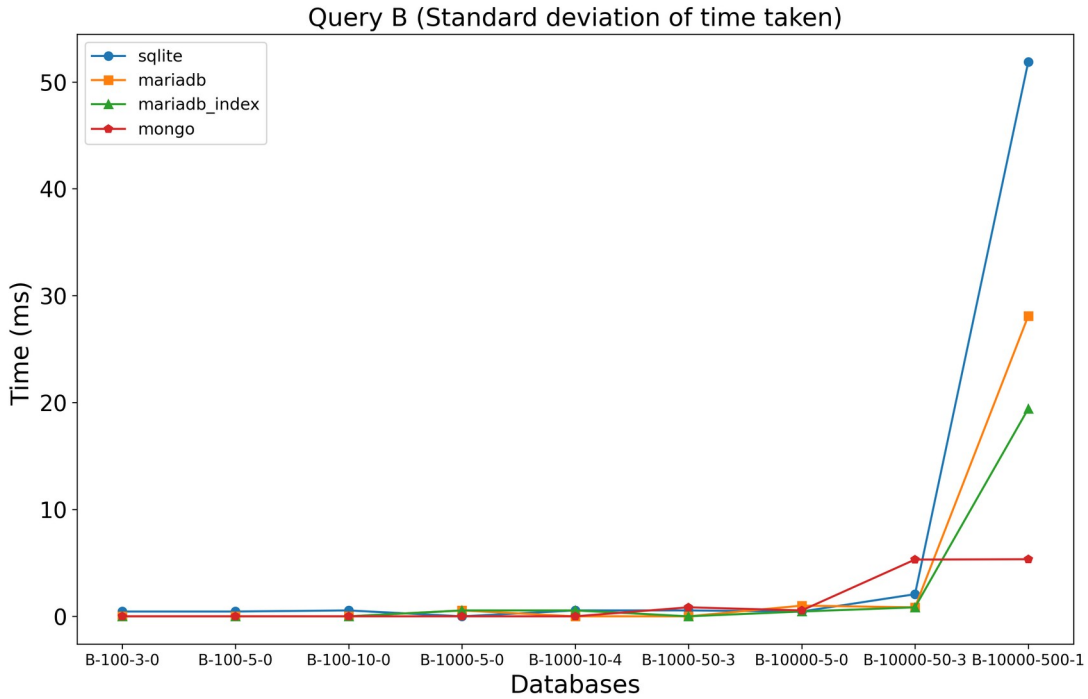




2) For query B, time taken by each engine on each database

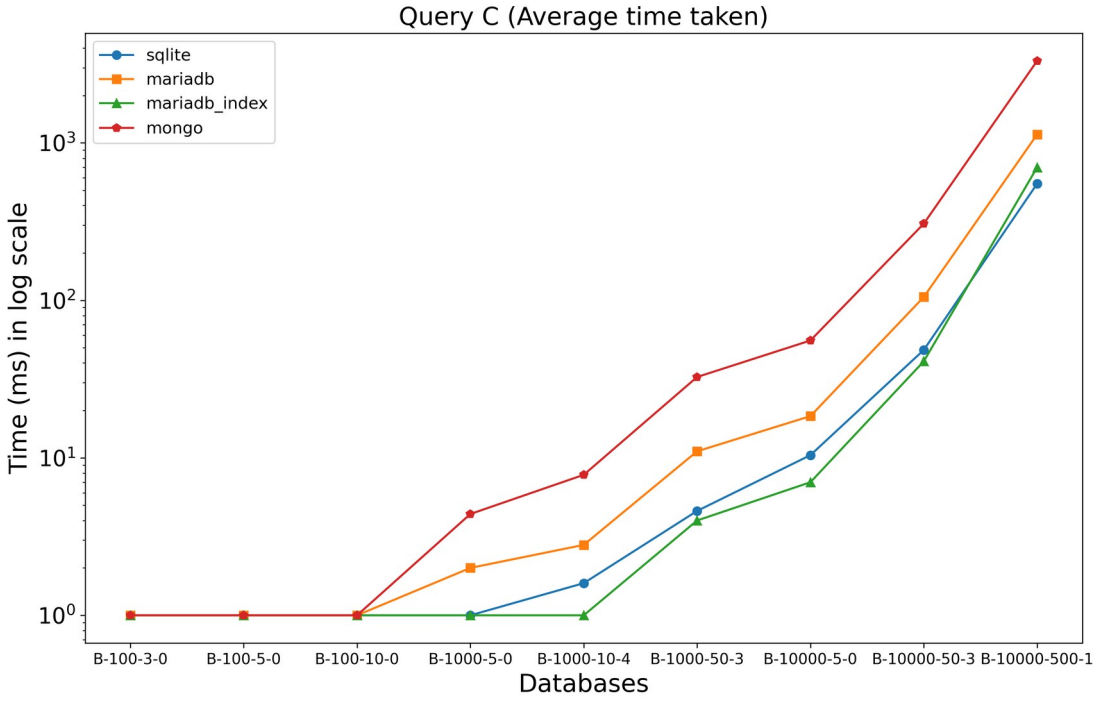
for plotting average time, used log axis for time

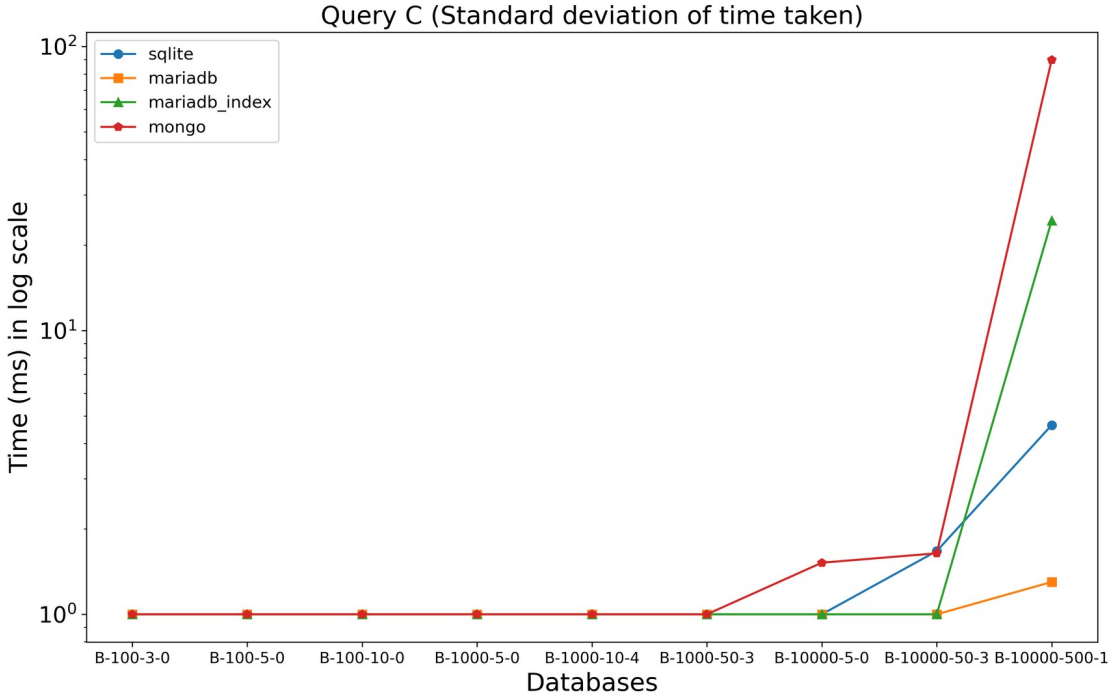




3) For query C, time taken by each engine on each database

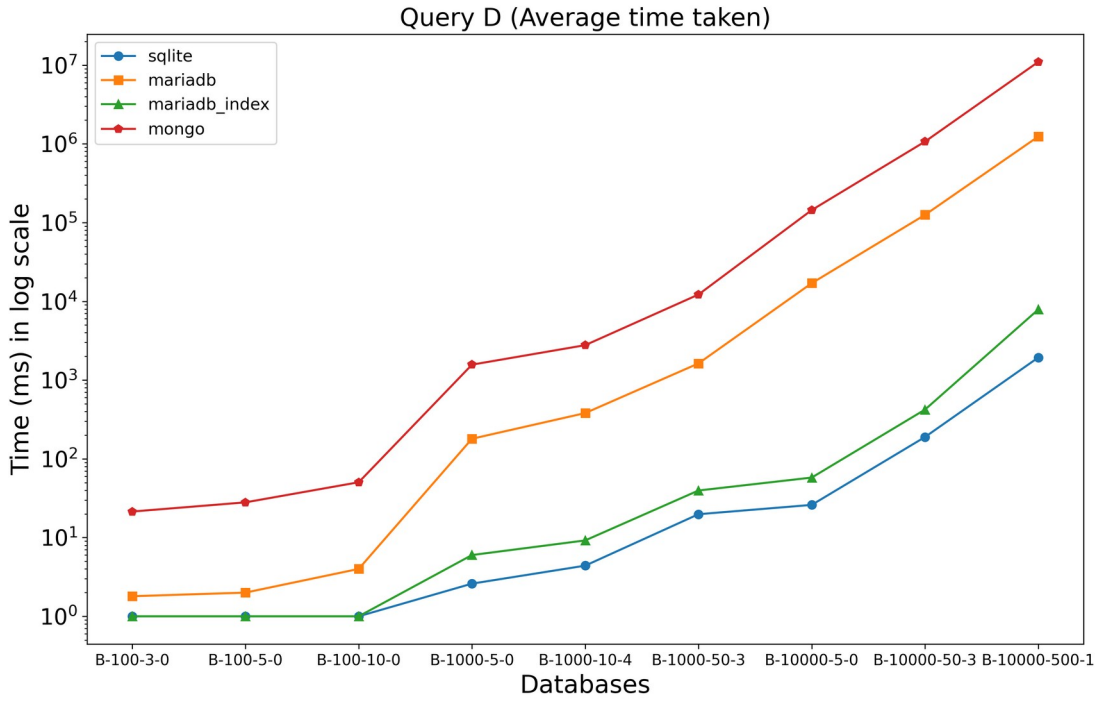
for plotting average time and standard deviation used log axis for time

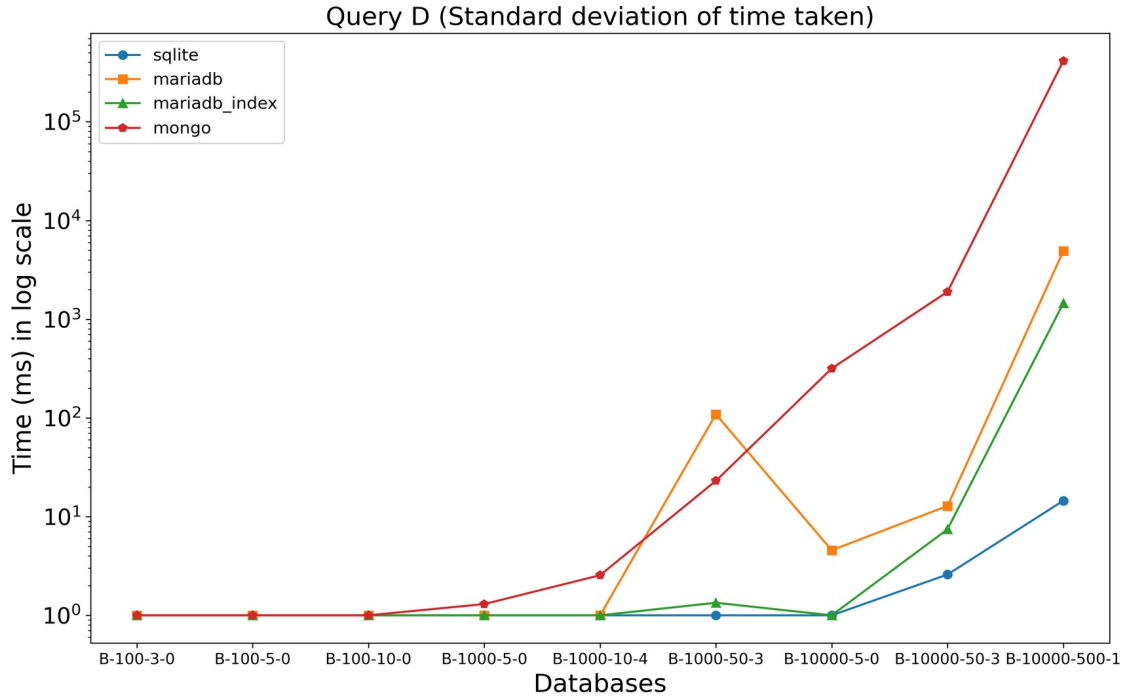




4) For query D, time taken by each engine on each database

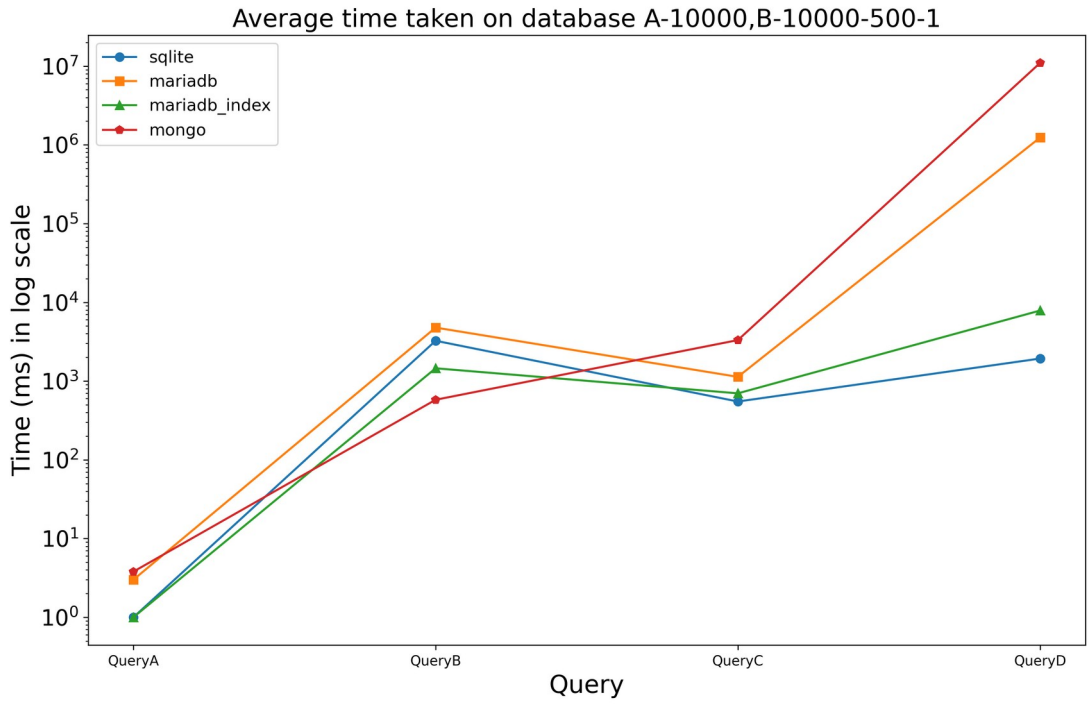
for plotting average time and standard deviation used log axis for time

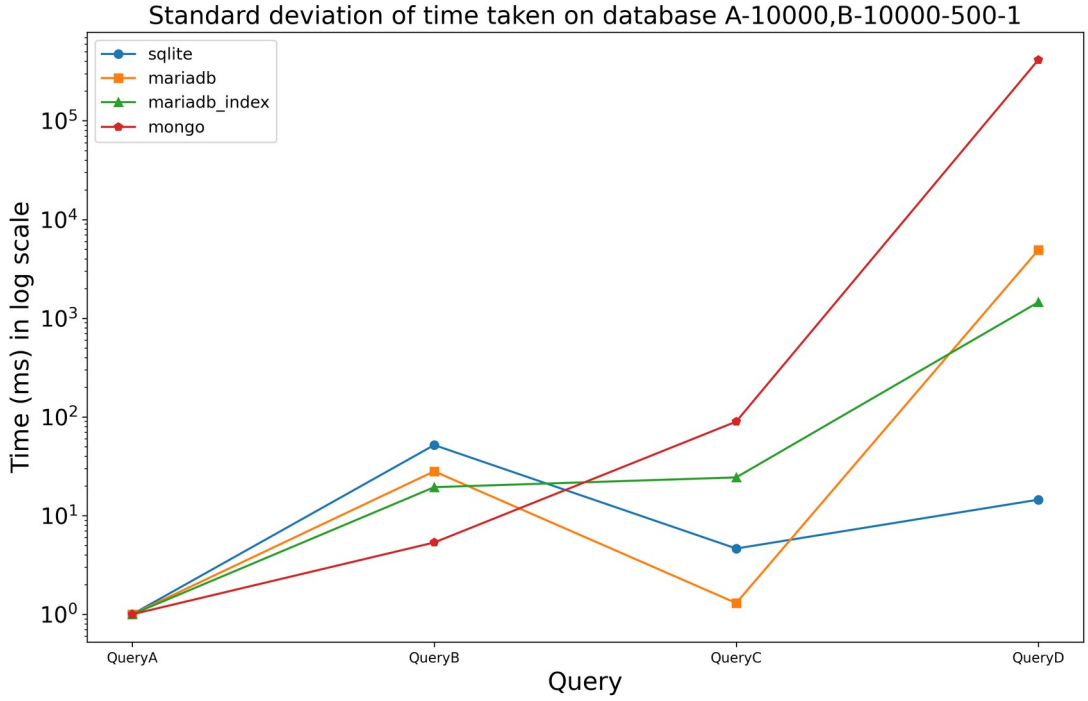




5) Comparing time for each query on each engine for database A-10000.csv, B-10000-500-1.csv

for plotting average time and standard deviation used log axis for time





4) Report

Machine Configuration

Operating system:	MacOS Mojave Version 10.14.6
Processor Name:	Intel Core i5
Processor Speed:	2.4 GHz
Number of Processors:	1
Total Number of Cores:	4
L2 Cache (per Core):	256 KB
L3 Cache:	6 MB
Memory:	8 GB 2133 MHz LPDDR3
Storage:	SSD 256 GB

Conclusion

Overview

From graph 1) and 5) we can conclude that query A takes least time where we find values in Table A for which A1 ≤ 50 because it is simple search query can be done using linear search or index. For A-10000.csv with 10,000 entries mongoDB and MariaDB without index takes more time because in MariaDB and MongoDB do not have index on A1.

For query B from graph 2), MariaDB with index takes less time than MariaDB without index and Sqlite because we created new index sort on B(B3 ASC, B1,B2) that improves the performance of sorting on B3. Indexing on MariaDb reduces time to about 1/4th when compared with MariaDB without index. From table of average time we can see that time taken by query is varying almost proportional to O(nlogn) where n is no. of entries in dataset. We also observe that for largest dataset B-10000-500-1 MongoDB gives least time shows that MongoDB handles big amount of dataset better than sql based engines.

For query C from graph 3), MongoDB takes most time for all databases because mongodb is unstructured database and group by operation is slower in mongoDB compared to sql based engines. MariaDB without index is slower than MariaDB with index on B2 because when MariaDB running on without index it is using file sort instead of index on B2. MariaDB without index takes almost double time than MariaDB with index.

For Query D from graph 4) we can see that query D takes most time and time taken significantly increases as size of data increases. MongoDB is slowest because it does not support join operation and use lookup which is very slow. For MariaDB, removing index on A1 and B2 significantly hurts performance this time for eg. in last database MariaDB with index takes around 7.8s while MariaDB without index takes around 20.6 minutes. Hence indexing improves the performance on Join operation by huge margin.

From graph 5) we can see that selection operation (query A) takes least time while Join operation is takes most time especially for MongoDB. For SQL based engine group by operation takes less time than Order by while for MongoDB sorting takes less time than group by because aggregate is slower in MongoDB.

Database engines

Indexing in MariaDB significantly improves performance especially in Join Operation

In our case, Sqlite shows almost similar or better performance than MariaDB with index except in query B (as we create new index on B3 on B in MariaDB which is not by default in sqlite). Because sqlite uses a very simple algorithm which is fast but does not handle concurrency whereas MariaDB handles concurrency and sqlite is file database which faster than socket based MariaDB. Hence for personal testing sqlite is better but for production MariaDB with index is better because it can handles multiple user

For standard deviation, sqlite shows least ratio of std dev to average time taken compared to MongoDB, MariaDB. Reason might be MongoDB and MariaDB both are server based while sqlite is file based, accessing from server might create variance in databases.

On query B, as size of data is large mongoDB gives least time that shows MongoDB handles large unstructured database better than sql based engines. MongoDB performs poor on operation like join and group by because it is not a relational database like MariaDB and Sqlite.

Scalability

We can observe from graph that as the size of data increases time taken to complete query increases significantly. But as size increases different engines handles different operations well. Using index on MariaDB significantly improves performance as size increases. On Sorting query B, for large database MongoDB handles it best.

System Issues

We can observe from graph of standard deviation that as time taken by query increases variation in time also increases because system handles multiple processes and as time increases it de-schedule and reschedule process multiple times which increase variation in time taken by query in different iterations.
And also when we produce results of queries it's output is either redirected to file or console which creates an I/O interrupt and effects time taken by query

5) Scripts

To get time for all queries on all databases
→ copy dbs folder in this folder (dbs folder must be in folder that has all scripts file)
→ run

```
chmod +x *.sh
./main.sh output
```

now time for all queries will get stored in folder **output**
final_output has output for all query that I got on mine machine
I have run each query 7 times for each database and stored there time in files mongo_A.txt, mariadb_A.txt, etc.

For generating graphs and table
→ install python3 libraries (matplotlib, pandas, dataframe_image)

```
pip install -r requirements.txt
```

→ run (it will create using files in output folder)

```
python3 clean.py output
```

→ to generate graphs on time that I got in my machine

```
python3 clean.py final_output
```

it will create folder **graphs** with graphs and table
final_graphs has graphs and table that I got on mine machine

script main.sh call script maria.sh, maria_index.sh, sql.sh, and mongo.sh on each databases for my roll no.
clean.py takes the output time calculate mean and average and plot graphs and table using matplotlib and pandas.