CS315A: Principles of Database Systems

Assignment 1

Guntas Singh Brar
180274

April 3, 2021

Databases

$$a = 2, \quad b = 7, \quad c = 4$$

$$(a \times a)\%5 = 4, \quad (a \times b)\%5 = 4, \quad (a \times c)\%5 = 3$$
$$(b \times a)\%5 = 4, \quad (b \times b)\%5 = 4, \quad (b \times c)\%5 = 3$$
$$(c \times a)\%5 = 3, \quad (c \times b)\%5 = 3, \quad (c \times c)\%5 = 1$$

- **db_100_1:** (A-100.csv, B-100-3-4.csv)
- **db_100_2:** (A-100.csv, B-100-5-4.csv)
- **db_100_3:** (A-100.csv, B-100-10-3.csv)
- **db_1000_1:** (A-1000.csv, B-1000-5-4.csv)
- **db_1000_2:** (A-1000.csv, B-1000-10-4.csv)
- **db_1000_3:** (A-1000.csv, B-1000-50-3.csv)
- **db_10000_1:** (A-10000.csv, B-10000-5-3.csv)
- **db_10000_2:** (A-10000.csv, B-10000-50-3.csv)
- **db_10000_3:** (A-10000.csv, B-10000-500-1.csv)

1. Query Problems

1. Find all A with $A_1 \leq 50$
2. Find all B in sorted order of B_3.
3. Find average number of values per A_1 by using only B table
4. Find all A_2 that corresponds to B by using B_2 (output the fields of B and A_2)

SQL Queries

1. SELECT * FROM A WHERE A1<=50;
2. SELECT * FROM B ORDER BY B3;
3. SELECT avg(X) FROM (SELECT count(B1) AS X FROM B GROUP BY B2) AS T;
4. SELECT A2, B1, B2, B3 FROM A INNER JOIN B ON A.A1=B.B2;

MongoDB Queries

1. db.A.find({A1:{$lte:50}})
2. db.B.aggregate([{$sort:{B3:1}}])
3. db.B.aggregate([
   {$group:{_id:"$B2", total:{$sum:1}}},
   {$group:{_id:null, avgVal:{$avg:"$total"}}}
   ])
4. db.B.aggregate([
   {$lookup:{from:"A", localField:"B2", foreignField:"A1", as: "Ax"}},
   {$replaceRoot:{newRoot:{$mergeObjects:[{$arrayElemAt:["$Ax", 0]}, "$$ROOT"]}}},
   {$project:{Ax:0}}
   ])

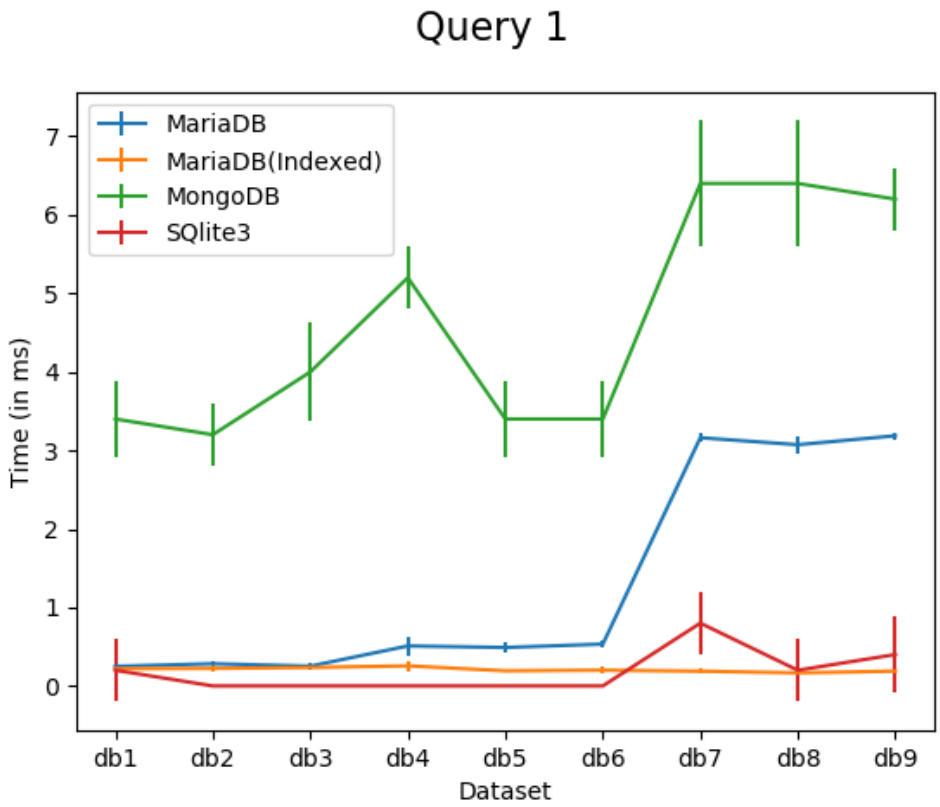
2. Query Time Data

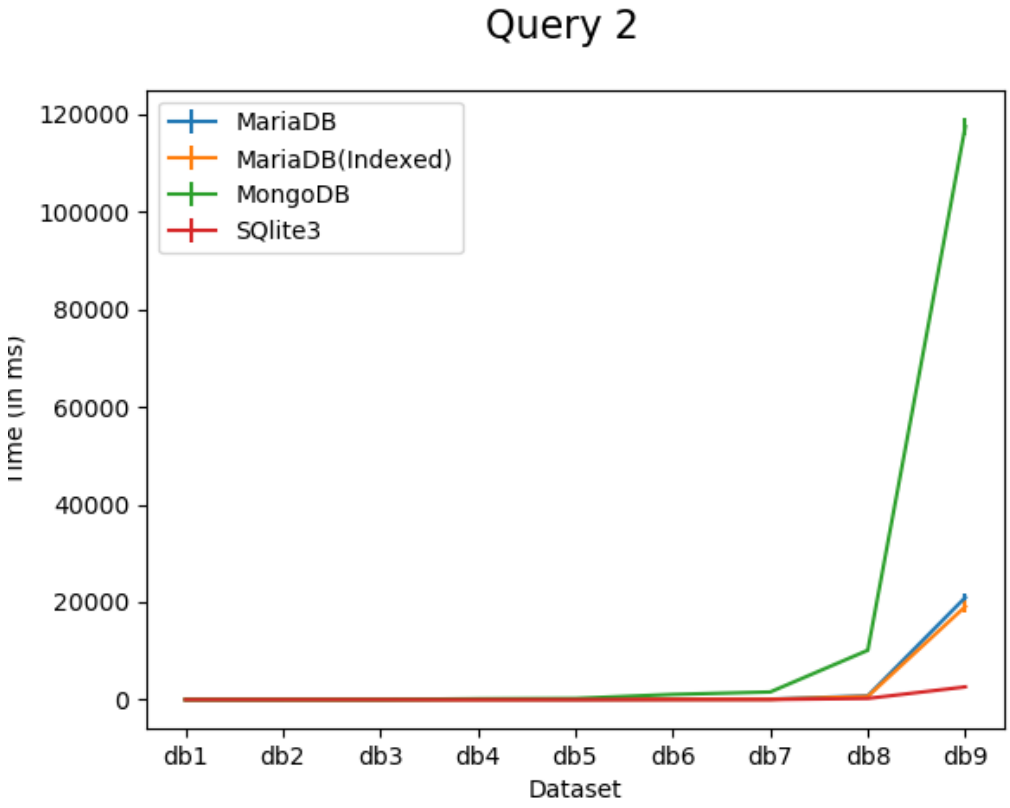
			db_100_1	db_100_2	db_100_3	db_1000_1	db_1000_2	db_1000_3	db_10000_1	db_10000_2	db_10000_3
SQlite3	Query 1	AVG	0.2	0.0.	0.0.	0.0	0.0	0.0	0.8	0.2	0.4
		STD_DEV	0.400	0.000	0.000	0.000	0.000	0.000	0.400	0.400	0.489
	Query 2	AVG	0.4	0.0	0.2	2.2	4.0	19.8	26.0	245.8	2612.4
		STD_DEV	0.489	0.000	0.400	0.399	0.489	0.489	0.632	14.851	187.936
	Query 3	AVG	0.0	0.2	0.4	1.0	1.4	5.6	8.0	72.4	815.2
		STD_DEV	0.000	0.400	0.489	0.000	0.489	0.489	0.000	6.945	37.209
	Query 4	AVG	0.0	0.0	0.2	3.0	5.4	23.4	33.0	257.0	2932.8
		STD_DEV	0.000	0.000	0.400	0.000	0.489	0.489	0.632	17.697	215.840
MariaDB	Query 1	AVG	0.249	0.282	0.252	0.509	0.489	0.533	3.161	3.072	3.186
		STD_DEV	0.014	0.033	0.037	0.118	0.075	0.045	0.055	0.106	0.045
	Query 2	AVG	0.477	0.781	0.982	10.830	17.665	78.435	100.520	777.304	20952.597
		STD_DEV	0.037	0.089	0.144	0.653	1.112	9.834	3.799	5.669	882.48
	Query 3	AVG	0.409	0.548	0.705	2.977	3.695	13.638	19.357	103.740	1060.329
		STD_DEV	0.036	0.104	0.060	0.354	0.406	1.356	1.533	2.121	5.591
	Query 4	AVG	2.404	3.704	6.913	288.663	527.011	2263.680	26118.877	192215.622	1877608.083
		STD_DEV	0.082	0.322	0.276	17.011	23.912	79.432	19.007	612.844	2850.934
MariaDB(Idx)	Query 1	AVG	0.219	0.223	0.234	0.255	0.192	0.200	0.188	0.166	0.188
		STD_DEV	0.016	0.038	0.011	0.066	0.018	0.044	0.033	0.002	0.009
	Query 2	AVG	1.405	0.606	1.267	6.332	27.609	54.125	70.454	663.648	19094.319
		STD_DEV	0.341	0.105	0.125	0.636	2.189	0.305	3.209	52.128	1342.879
	Query 3	AVG	0.367	0.331	0.409	1.267	1.550	4.890	7.802	38.733	403.561
		STD_DEV	0.086	0.074	0.057	0.182	0.244	0.761	1.049	0.493	23.200
	Query 4	AVG	0.852	0.467	0.628	3.673	7.198	16.316	25.864	169.477	1717.848
		STD_DEV	0.210	0.063	0.061	0.507	0.803	0.007	0.056	13.263	134.603
MongoDB	Query 1	AVG	3.4	3.2	4.0	5.2	3.4	3.4	6.4	6.4	6.2
		STD_DEV	0.489	0.399	0.632	0.399	0.489	0.4898	0.799	0.799	0.399
	Query 2	AVG	10.0	14.4	28.4	181.8	249.2	1069.2	1572.6	10120.6	117473.4
		STD_DEV	0.0	1.496	3.440	11.178	18.755	27.952	127.278	97.518	1618.575
	Query 3	AVG	0.8	1.0	1.4	5.4	5.2	16.8	29.2	147.0	1141.6
		STD_DEV	0.4	0.0	0.489	0.489	0.399	4.791	0.399	3.633	41.735
	Query 4	AVG	26.0	37.0	69.4	1403.4	2135.0	9442.6	103440.2	742355.4	7391556.8
		STD_DEV	1.549	4.147	5.351	73.925	52.569	139.230	2508.063	23530.913	165210.411

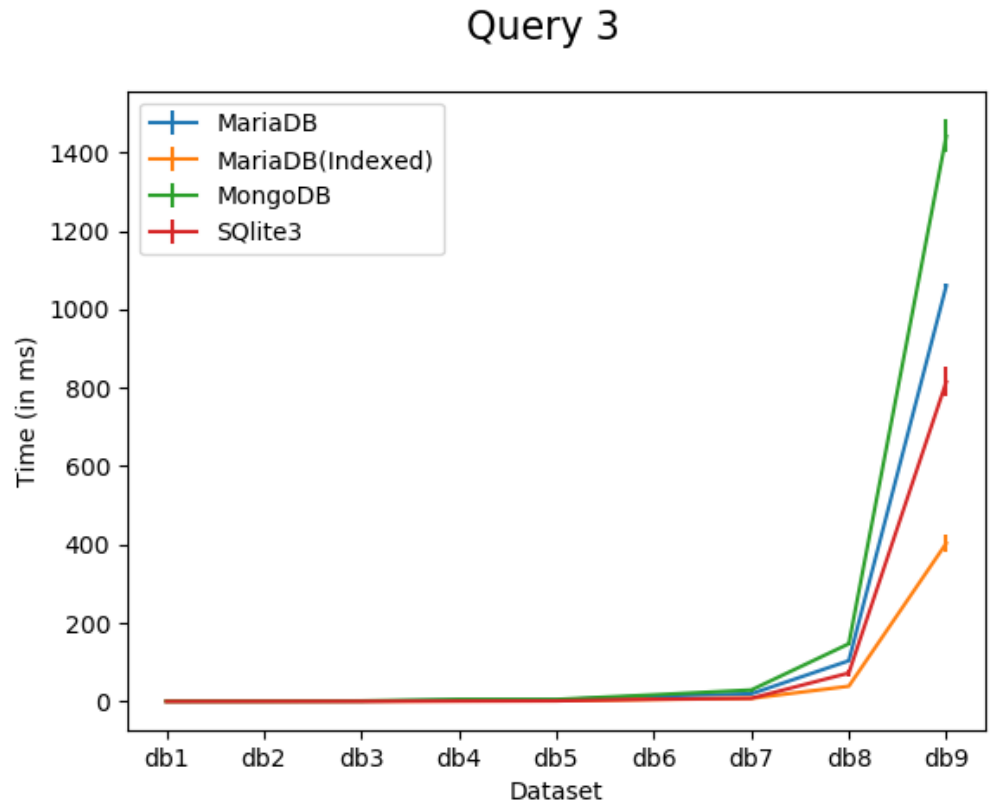
Table 1: Query time data for all 9 data sets on all 4 database systems (in milliseconds)

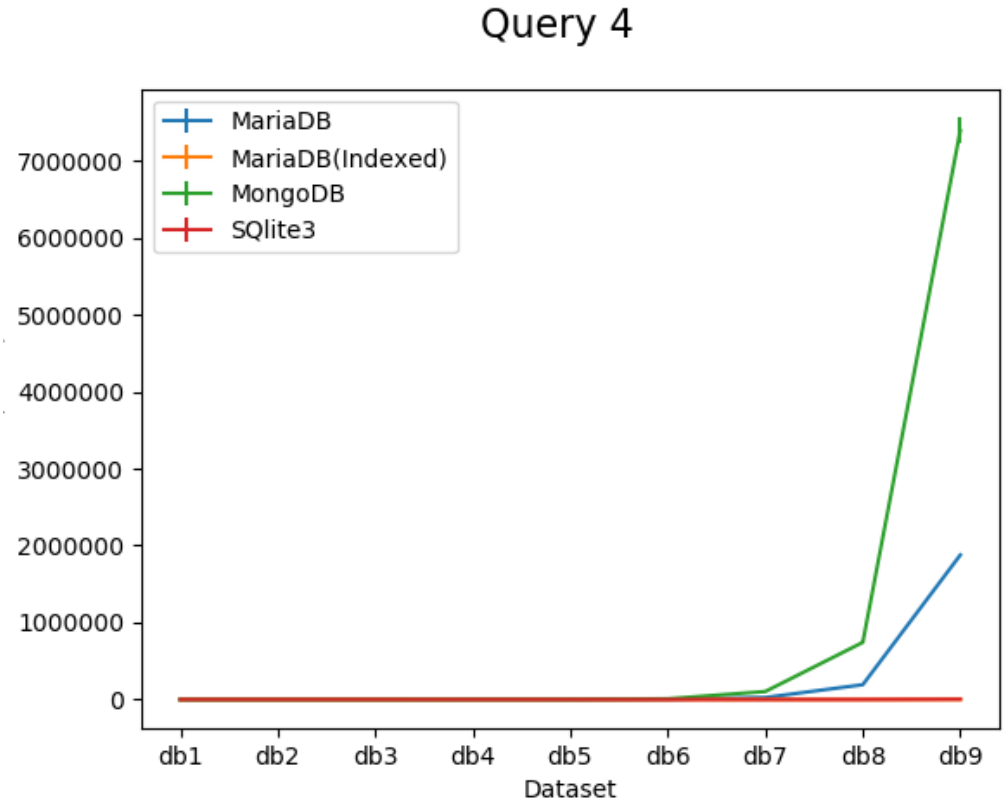
3. Graphs

Per Query

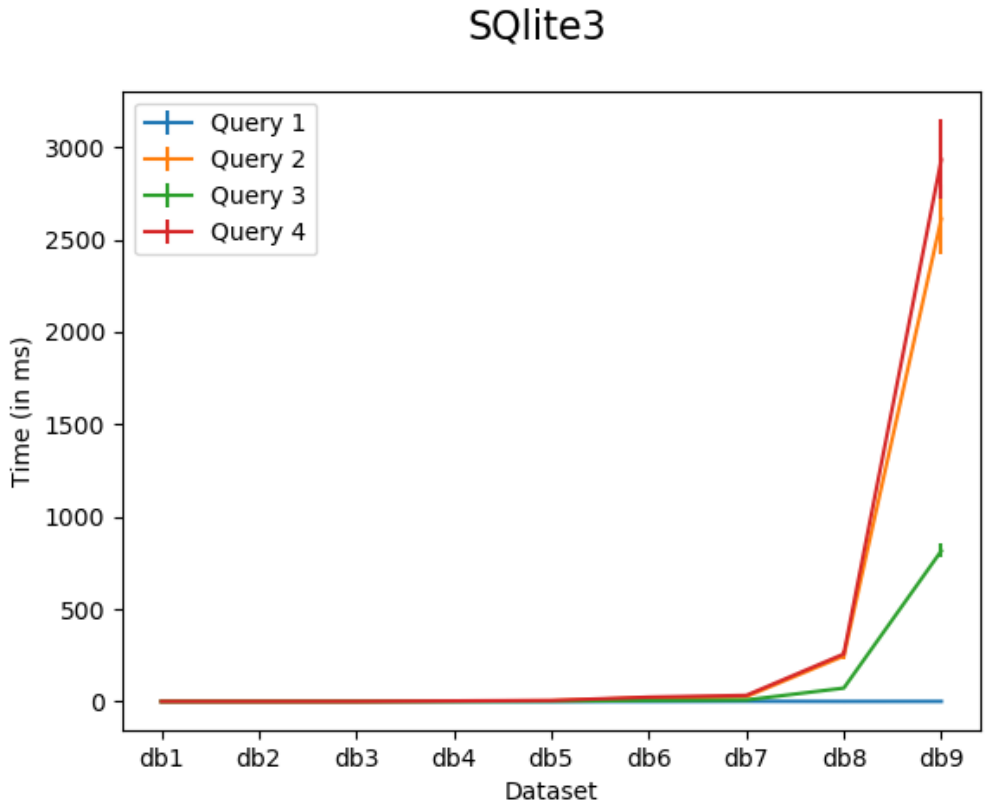


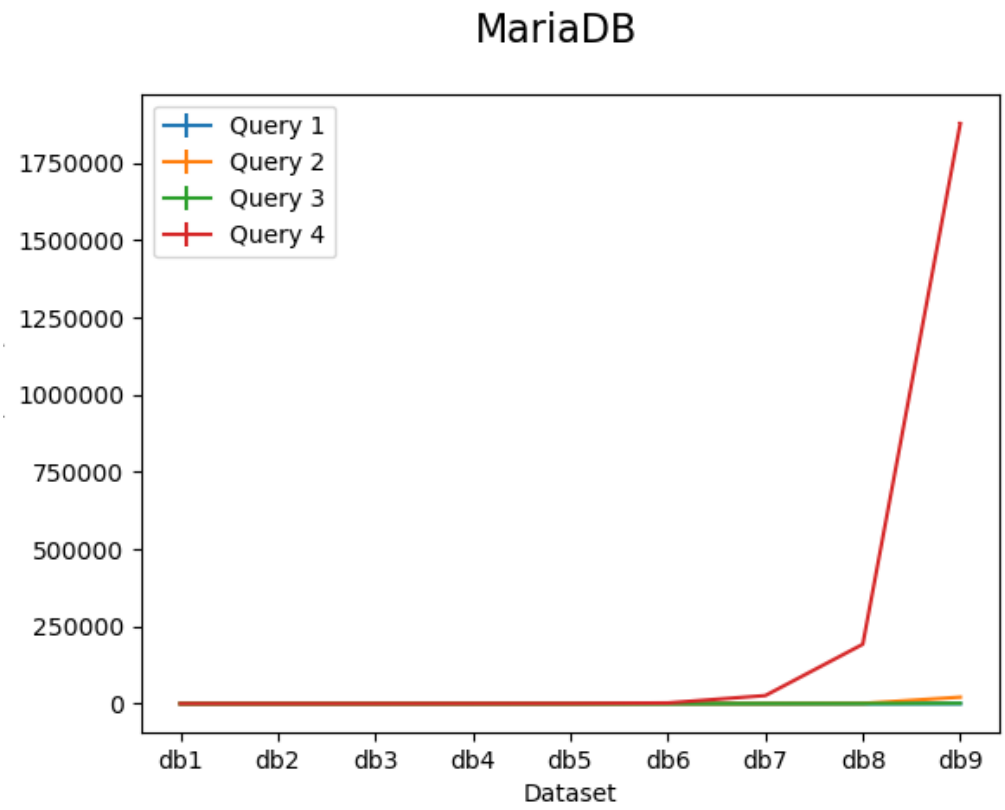


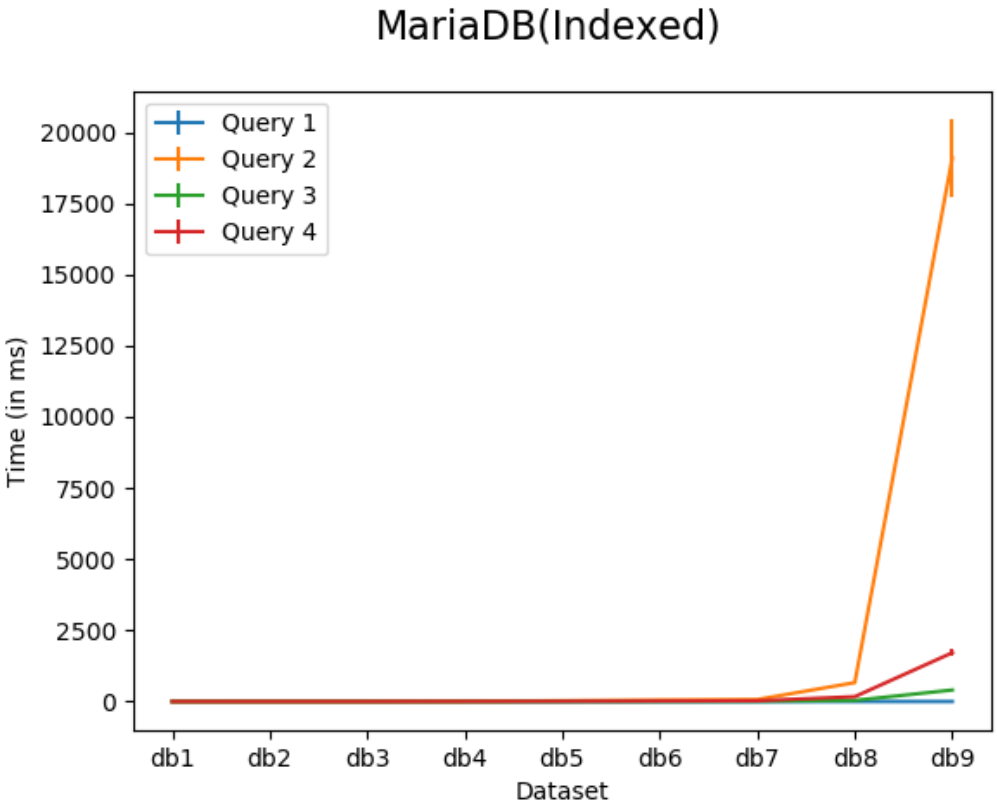

Query 3
MariaDB
MariaDB(Indexed)
MongoDB
SQlite3
Time (in ms)
0
200
400
600
800
1000
1200
1400
db1
db2
db3
db4
db5
db6
db7
db8
db9
Dataset

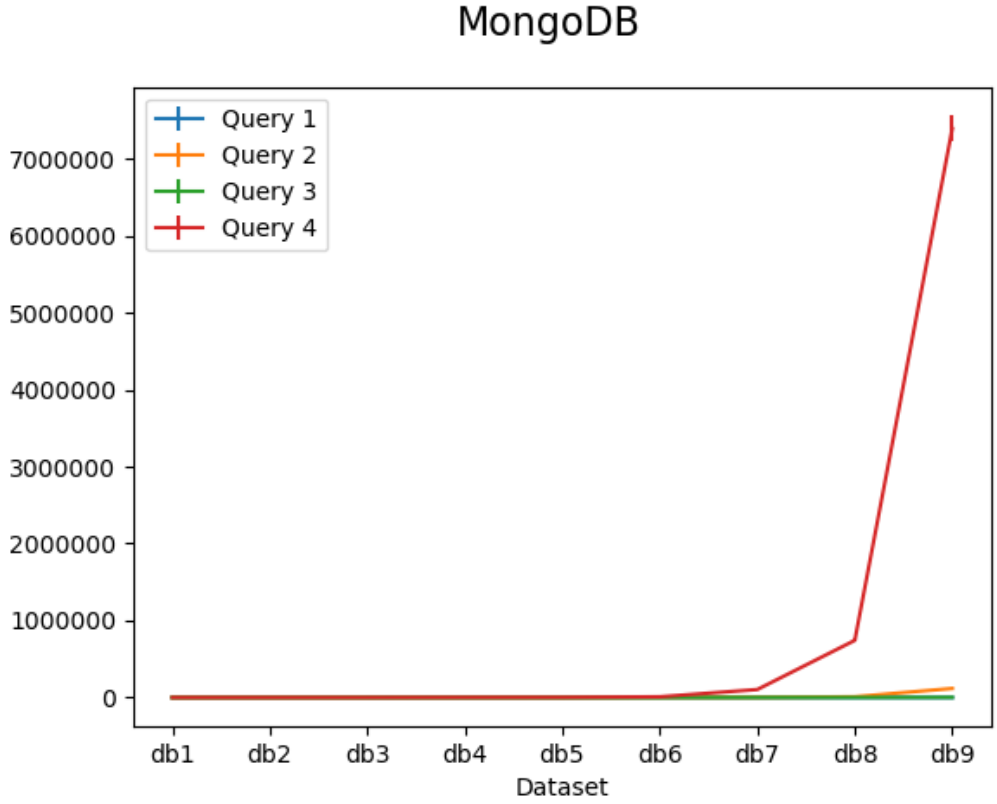

Query 4
MariaDB
MariaDB(Indexed)
MongoDB
SQlite3
0
1000000
2000000
3000000
4000000
5000000
6000000
7000000
db1
db2
db3
db4
db5
db6
db7
db8
db9
Dataset

Per Database System






MariaDB(Indexed)
Query 1
Query 2
Query 3
Query 4
20000
17500
15000
12500
10000
7500
5000
2500
0
Time (in ms)
db1
db2
db3
db4
db5
db6
db7
db8
db9
Dataset


MongoDB
Query 1
Query 2
Query 3
Query 4
7000000
6000000
5000000
4000000
3000000
2000000
1000000
0
db1
db2
db3
db4
db5
db6
db7
db8
db9
Dataset

4. Conclusions

From the data and the plots above, we can conclude the following:

- MariaDB with indexing is much faster than MariaDB without indexing for all queries. The differece is very significant and evident for the 4^{th} query.
- MariaDB with indexing is very fast as compared to the others on very large databases.
- NoSQL databses are very slow for operations like join. MongoDB takes a lot of time for queries involving join operations. For systems involving such queries, relational database systems must be used
- The previous point is also observed for relational databases if we do not implement any indexing. We can see that both MongoDB and MariaDB(without Indexing) struggle for join operations on large databases.
- MongoDB takes relatively very less time on sorting as compared to join.
- Query 2 and Query 4 take the most time out of the four queries as the former is sorting the table B so it has to go through all the tuples of B and the latter involves joining A and B.

Machine and OS Specifications

- **Operating System:** Ubuntu 20.04.1 LTS (64-bit)
- **CPU :** Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz
- **Number of Cores :** 6
- **Threads Per Core:** 2
- **RAM:** 16GB
- **Disk Space:** 100GB SSD + 100GB HDD

5. Scripts

- All the scripts used have been included in the submission.
- They are organised database system wise and their names are simple and understandable.
- There are three top-level scripts in the top directory to create, drop or query all the 4 database systems at one go.
- A python sctipt plot_data.py is in the top directory which processes all the data and plots the graphs(the ones in this report) into the plots/ directory.

IMPORTANT

- Do not delete any of the directory or change the directory structure. It may lead to numerous errors as the scripts are written according to the submitted directory structure.
- Copy the requierd csv files(mentioned at the beginning of the report) into the dbs/ folder before creating databases.
- The raw query output can be seen in the data/query_dump/ directory
- The time data extracted from query dumps can be seen in the data/query_time_data directory