Project 3: Advanced Performance Tuning

T-508-GAG2 Performance of Database Systems

Instructors: Björn Þór Jónsson, Grímur Tómas Tómasson

Teaching assistant: Patrekur Patreksson

March 27th 2019

Group name: Unicorns

Names:

Daníel Örn Melstað (danielm15@ru.is)

Halldór Bjarni Stefánsson (halldors17@ru.is)

Hera Hjaltadóttir (hera16@ru.is)

Martha Guðrún Bjarnadóttir (marthab16@ru.is)

Víðir Snær Svanbjörnsson (vidir17@ru.is)

Table of Contents

1. Introduction

After tuning the *innheimta* database (DB) and the provided queries in the previous project (Project 2), we wanted to build a new solution on top of the previous one and improve performance even further. For this project we were allowed to use all methods allowed in project 2, as well as the following:

- Change disk based tables to memory based ones
- Perform SQL in natively compiled procedures
- Compress data
- Use columnar indices

Since many changes were made in Project 2 using the previously allowed methods, we mainly focused on these new additions. The methods and their effects on performance will be outlined in chapter 3, along with a brief discussion on methods what we tried that didn't improve performance. In chapter 2 we describe the technical environment used for the project and in chapter 4 we discuss our results and draw our conclusion.

Overall we noticed the largest reduction in query execution time by adding columnar indices. We were able to slightly improve performance by adding more indices suggested by the Database Engine Tuning Advisor and some of the statistics we noticed were missing in the execution plans of the queries.

2. Technical environment

In order to keep measurements consistent and accurate, they were all made on the same computer and software environment.

2.1 Hardware

All of the tests were run on an Acer Nitro AN515-51 laptop, the specs for this computer are the following:

CPU: intel core i5 7200U 2.5GHz, 2701Mhz, 2Core(s), 4 Logical Processor(s)

RAM: 8 GB DDR4.

Storage: INTEL SSD MODEL SSDPEKKW256G7 with 237 GB storage.

2.2 Software environment

The experiment was performed with the following technical environment:

Server:	Microsoft SQL Server Enterprise 2017
Program:	Microsoft SQL Server Management Studio
OS:	64-bit Windows 10

3. Results

In order to find out which changes should be made to the database, we tried different things and kept any changes that improved performance and scrapped any changes that decreased performance.

3.1 Changes made

We first tried changing some of the largest tables in the DB from disk based tables to memory based, but found it lessened performance. We then moved on to try using stored procedures, but after trying different applications of it we didn't find it to improve performance for us. Compressing different tables did not result in any improvement either. Only when we started adding clustered columnstore indices on the tables *hreyfingMain* (a partition from Project 2 of *hreyfing*), *vidskiptamadur* and *krafa* we got better results.

After seeing no improvements using the newly introduced methods listed in Chapter 1 except for the columnstore indices, we tried our luck with the methods used in Project 2. By adding some of the statistics suggested by the query execution plan, we were able to get a slight improvement in performance. Interestingly though, we found that some of the suggested statistics made performance worse. We also tried many of the indices suggested by the DB Engine Tuning Advisor and found some of them improved performance.

3.2 Final results

Our baseline measurements are simply the final measurements from Project 2. To measure the performance of the improved database, each query was measured 5 times for each significant change made, the highest and lowest time discarded and the average of the remaining three calculated. For the timing process a timing script was provided that conveniently measures the speed and number of I/Os of the scripts and puts them neatly into an easily readable table. All of the measurements were made using the 'final version' of the timing script provided.

Query	Baseline CPU time (ms)	Baseline Wallclock time (ms)	New CPU time (ms)	New wallclock time (ms)	Wallclock time improvement ratio (new:old)
Aggregate	13773.07	6657.53	3494.65	3995.96	0.60
Insert	0.03	0.03	0.05	0.05	1.53
Update	2469.51	3688.79	2011.43	1214.82	0.33
Scenario 1 ID 1	6978.09	4758.76	7586.48	5497.66	1.16
Scenario 1 ID 10	3629.14	1390.52	4472.94	2245.64	1.62
Scenario 1 ID 40	3085.52	1034.75	4034.78	1877.19	1.81
Scenario 1 ID 200	3017.73	961.63	3907.48	1778.14	1.85
Scenario 2 ID 1	3197.14	2967.26	4469.63	5860.94	1.98
Scenario 2 ID 10	2094.00	2224.25	3397.33	2310.67	1.04
Scenario 2 ID 40	1076.81	1566.85	2385.62	1567.54	1.00
Scenario 2 ID 200	771.62	1408.33	2058.61	1316.85	0.94
Scenario 3 ID 1	5382.92	4806.25	2532.06	1735.22	0.36
Scenario 3 ID 10	4264.65	4287.39	2127.17	1431.71	0.33
Scenario 3 ID 40	3807.08	4350.17	2098.52	1334.46	0.31
Scenario 3 ID 200	4571.29	4134.25	2061.73	1340.35	0.32
Total (all queries)	58118.60	39886.59	46638.48	33507.20	0.84

Table 1: The baseline times are from the measurements made after all the changes made in Project 2. The new time measurements are the results of the improvements made for the current project.

As can be seen by the improvement ratio in Table 1, some of the queries performed worsen with our improvements. However, as we see by summing up the total time taken for all queries, there is an overall improvement that decreases the total time by 6379.39 milliseconds which amounts to a 16% decrease. This is not a big improvement and we were hoping to see a larger difference, but we did try many different things that we theorized might improve performance but found out it didn't improve or made it worse.

3.3 Further improvement attempts

After gaining some experience from the previous projects, we had some ideas on how we could further improve performance.

We wanted to reduce usage of the table *postnumer*, so we tried making a new table, *heimilisfang_new*, containing the same information as *heimilisfang* and *postnumer* combined, with the exception of the column *heimili_2*, which was never used and its contents looked like utter nonsense. Much to our surprise and disappointment, this did not improve performance.

Next we noticed that there were several calls to `ISNULL()` to get addresses from *thjodskra* where they were missing in *vidskiptamadur* and *greidandi*. To eliminate this we tried to insert the missing addresses into *vidskiptamadur*. This attempt failed miserably and caused the queries to return incorrect results.

The last thing we tried was to partition the second largest table *krafa* based on status ('Greidd', 'Ógreidd' or 'Niðurfelld'), since we were often only fetching entries of one status. First we created the table *krafaGreidd*, which improved the performance of the aggregate query but had no effect on nonaggregate scenario 1. Then we created *krafaOgreidd*, but this decreased the performance of nonaggregate scenario 3 significantly. We also predicted that the effect on nonaggregate scenario 2, which needs all statuses, would be negative and so we abandoned our efforts.

4. Conclusion

In this project we improved the performance of the DB and queries from project 2. We found out that columnstores can be a good addition to a DB to make queries run faster, in our case it caused an improvement of 16% which is a good addition to the already optimized system. We were surprised to see how many things that we intuitively thought could improve performance did not have the desired effects. Tuning databases is not a one-size fits all process and methods need to be experimented with and applied on a case-by-case basis. What might work in theory or improves performance in certain situations might have a detrimental effect on performance in other cases.