The Implementation of Caching Database to Reduce Query's Response Time

Article *in* MATEC Web of Conferences · May 2016

DOI: 10.1051/matecconf/20165803014

Arnaldo Sinaga
Institut Teknologi Del, Indonesia
23 PUBLICATIONS **80** CITATIONS

DOI: 10.1051/matecconf/20165803014

THE IMPLEMENTATION OF CACHING DATABASE TO REDUCE QUERY'S RESPONSE TIME

Arnaldo Marulitua Sinaga[1], Poppy Sibarani[2]
[1,2]Informatics Department, Faculty of Electronics and Informatics Engineering, Institut Teknologi Del, Indonesia
E-Mail: [1]aldo@del.ac.id

ABSTRACT

Performance is an imperative aspect of a web-based application. Response time is one of the most important performance parameters. Most of web-based applications are driven by database system. Reducing response time of its database system will improve the performance of a web-based application. Caching is one of techniques that can be implemented to reduce response time. The implementation of caching can be conducted in the application or database side. Caching is a technique to keep the executed query and hence the query will not be executed when called in the later execution. The result of an executed query will be stored in the cache therefore when that query is called, the database will retrieve the stored result without re-executing query. This mechanism will reduce the time to re-execute queries. However, this technique may be not suitable for all queries. This research is intended to investigate the effect of the implementation of caching into performance of a database. A series of experiments have been conducted by applying query cache. The response time of the application of query cache is compared with the response time of database without query. The results show that the implementation of query cache reduces response time significantly. However, this technique is more suitable for large database with many frequently asked queries. The number of clients accessing the database is found to be influencing the performance of caching. The more clients access the database, the more improvement is provided by query cache.

Keywords: database, caching, query, performance

INTRODUCTION

Many web-based applications are built using a multi-tier environment (Larson et al., 2004). Multi-tier client environment involves the use of client-tier, middle-tier and database-tier. Client-tier is the part that relates directly to the client, one of which is the browser. Through a browser, the client can connect with middle-tier. Middle-tier components relate directly to the database tier. Middle-tier contains business process that serves to process client requests that come through the client-tier to retrieve the data in the database-tier. Middle-tier itself can be divided into several types in use, such as web-based application or desktop-based. By the time the client is using a browser, an application that is used is a web-based application consisting of a web server and web client. Results from the web client request will be returned through the web server.

Each user who accesses the application will make a request through the browser, which is then forwarded to the web server. On the web server, the request is processed and routed to the database. Then, the application returns the response to the user via the browser. Web servers always retrieve data from the database to process each request that comes, regardless of whether a new request comes together with a previous request. The heavy workload of web servers can cause a problem into the network traffic (Qiong et al., 2002, Charles et al. 2008, Paweł and Krzysztof, 2011).

Caching is used to overcome the workload problem including Web Caching (Jesse et al., 2002), Web Databases Caching (Alexandros, 2010) and Database Caching (Qiong et al., 2002, Charles et al. 2008, Paweł and Krzysztof, 2011). This research investigates the affect the Query Cache into Database. Database Caching is an imperative method to overcome the problem in high-traffic web-based applications (Priya et al., 2011). With the implementation of Query Cache, is expected to reduce the response time in executing the query. On applying the Query Cache database, user queries are executed in cache on the local database or the database contained on the database server.

This research is intended to examine how the effect of the application of caching in the database of the performance, especially in query response time. From this study, it is expected to obtain the guidance of applying caching in database.

This research is conducted by using empirical approach. A series of experiments are held that is by applying the Query Cache in a subject application.

LITERATURE REVIEW

A. Performance Tuning Overview

Database performance tuning consist of three steps: Performance Planning, Instance Tuning and SQL Tuning. Performance tuning needs to be well planned involving all resources. Maintenance is important to decrease the problem in performance. Good management of all components of database in a system will reduce or avoid the problems during operation (Immanuel and Lance, 2014). Instance Tuning is involved when a database initialized that is to prevent bottlenecks. In the initial configuration of a system, resource allocation will be involved. The most important of instance tuning is to identify bottlenecks and make appropriate changes to eliminate the problem of the bottleneck. Tuning is usually done repeatedly, that is when the system starts to operate or while running. In general, tuning implement improvements in performance problems. In other words, tuning should be the part of the life cycle of an application system. Typically, tuning phase will be ignored until the database is generated. Tuning can be a reactive process, where most major bottleneck problems will be identified and repaired (Immanuel and Lance, 2014).

When the SQL statements executed on the database, query optimizer will determine the exact execution plan and factors relating to the object as well as the conditions specified in the query. The main purpose of query optimizer is to process the SQL statement and reduce the execution time. Throughout the process of execution, the query optimizer statistics gathered and used in reviewing the system to determine the best address data access and other considerations. We can reuse the (override) execution plan of the query optimizer by giving different input into the SQL statement.

B. Caching

Caching is a technique that can be used to improve the performance of a system. The cache can be interpreted as part of a computer's memory where information is stored so that the computer can find the information faster (http://www.merriam-webster.com/dictionary/cache, 2014). The stored information can be a result of the previous computation process or a duplication of the original information. There are some caching techniques that can be done to improve the performance of web-based applications that implement multi-tier concept. Caching can be performed on the database tier, the middle-tier and the client tier.

Caching technique in MySQL is named as Query Cache. The architecture of MySQL is shown in Figure 1.

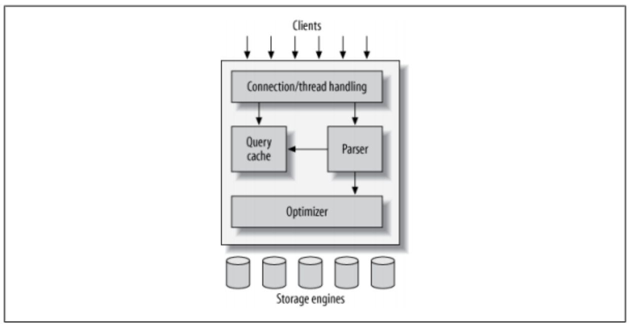

Figure-1. The Architecture of MySQL (Schwartz B. et al., 2012)

The upper layer of MySQL architecture contains one or more services that are needed by client/server (Schwartz B. et al., 2012), such as connection handling, authentication, security and so on. Important processes are held in the second layer, including code for query parsing, analysis, optimization, caching and all built-in function. The third layer contains storage engine that responsible for storing and retrieving data. The followings are steps of query execution (Schwartz B. et al., 2012):

1. Before parsing a query, MySQL checks whether Query Cache is active or not. If active, MySQL checks if the query has been stored in cache. MySQL the checks the privilege, if allowed MySQL returns Query Cache results to the client. When query is not in the cache, Query Optimization process will be activated to obtain the result of the query.
2. In Query Optimization process, MySQL parser divides a query into tokens and builds parse tree of those tokens. Parser ensures that the token is valid, and every part is in proper order.
3. Preprocessor checks related tables and columns and make sure the referenced name or alias is not ambiguous. Furthermore, the preprocessor checks privileges.
4. After the process of the parser and preprocessor, parse tree is ready to be converted into execution plans by the optimizer.

The query is executed on the Query Execution Engine in accordance with the execution plan that has been generated previously.

Query Cache is a technique that can be performed on systems with multi-tier concept. The purposes of applying the Query Cache is to increase performance and scalability of applications with distributed query processing (Qiong et al., 2002). At the time of Query Cache is placed on the middle-tier along with the application server, the Query Cache works like a database server. Query Cache can keep most of the existing data on the database server (Qiong et al., 2002). Thus, the server workload can be shared on the Query Cache. With the division of the workload of the database server, the response time to the user can be reduced, especially when the database server does the heavy work (Qiong et al., 2002).

To ensure the beneficial use of Query Cache, server operations have to be tested both with the cache on and off (http://dev.mysql.com/doc/refman/5.5/en/query-cache.html, 2014). The efficiency of Query Cache can be changed when the workload of the server is changed.

Query to be executed is compared with the correspondent query in Query Cache (http://dev.mysql.com/doc/refman/5.5/en/query-cache-operation.html, 2014). They have to be exactly the same. The value of variable Qcache_hits is incremented each time the query cache is being used. When a table has been changed/updated, all related queries in the cache will be deleted including all queries that have join operation with that deleted table.

MySQL stores query cache in the memory. The stored information in query is not just the result set, but also all the structure of the query including the relation between all involved tables with its query result and query text. Besides the housekeeping (base structure), query cache memory pool is provided in variable-sized block (Schwartz B. et al., 2012). When server is run, memory for query cache is initialized. Memory pool is initialized as an empty single block. When the server caches a result set, the size of allocated block in memory pool is set to query_cache_min_res_unit (Schwartz B. et al., 2012).

There are some constraints of Query Cache. Query Cache is not applicable to the following cases: query is a sub-query, the executed query is in a function, trigger or event, referring to the user or local stored program variables, referring to INFORMATION_SCHEMA, or PERFORMANCE_SCHEMA, referring to partition table, query uses temporary tables, query without table and user without any privilege to access the correspond tables (http://dev.mysql.com/doc/refman/5.5/en/query-cache-operation.html, 2014).

Query Cache store the text of the SELECT statement and the results of the statement are sent to the client (http://dev.mysql.com/doc/refman/5.5/en/query-cache-in-select.html, 2014). If the server receives the same statement/query as the previous statement, then the server will take the result of Query Cache and will not re-execute the statement. Query Cache is divided into sessions, so that a result set resulting from the query execution carried out by a client, can be reused as a response to another client (http://dev.mysql.com/doc/refman/5.5/en/query-cache-in-select.html, 2014).

Data stored in the cache is always the last data and trustworthy. All statements INSERT, UPDATE, DELETE or other modifications at a table will make the relevant data on the Query Cache eliminated (flushed) (http://dev.mysql.com/doc/refman/5.5/en/query-cache.html, 2014).

THE MATERIAL AND METHODOLOGY

This research is conducted by using empirical approach. A series of experiments are held that is by applying the Query Cache in a subject application. Del's library database system is used for the experiments. It is analyzed in order to design a caching application on the database. Data and running queries on the database will be reviewed if it is adequate for use in this study. Dummy data and the query will be added to improve the accuracy of the results.

A web-based application is used to simulate the application of the studied method. There are two types of environment being applied: standalone and client-server system. The experiments performed on a standalone system involve the database server and web server. Application on a web server executes queries in the database server. While the client-server system involves five clients that simultaneously execute queries in the database server. The response time of query execution with caching and without caching will be compared in both environments. The less response time the better method.

There are 17 queries involved in the experiments. Those queries use three tables in the database. Table 1 lists the used tables with their size. There are four phases (P1, P2, P3 and P4) of implementation that are intended to observe the influence of the size of the database. The phases are respectively involving the original tables, tables with 200% additional records of P1, tables with 300% additional records of P2, and tables with 400% additional records of P3.

Table-1. The size of the used tables

Table	Size (MB)			
	P1	P2	P3	P4
Books	1.57	4.72	16.25	76.13
Members	0.16	0.43	2.26	8.91
Book_issues	8.91	23.59	84.62	415.45

The selected queries are the frequently used queries in the library system. All queries uses SELECT operation and involve book_issues table. Queries can be classified as follows:

1. Q2 and Q7 are queries that involve a table with a condition in the WHERE clause.
2. Q1, Q8, Q10 Q11 Q14 and Q17 are queries that use a mathematical operation functions (COUNT and MAX).
3. Q5 and Q12 are queries that involve two tables and specific conditions in the WHERE clause.
4. Q15 is a query that involves two tables and two conditions on the WHERE clause.
5. Q3, Q4 and Q13 are queries that involve three tables with two conditions on the WHERE clause.
6. Q6 is a query that involves three tables and a condition in the WHERE clause.
7. Q9 and Q16 are queries that involve two tables and four conditions in the WHERE clause.

RESULTS AND DISCUSSIONS

In this section, the results of the experiments are presented. Table 2 presents the response time of query execution in the first two phases (P1 and P2) of standalone system.

Table-2. The result of P1 and P2 in standalone system

	P 1		P2	
Query	Non-Cache	Cache	Non-Cache	Cache
Q1	0.0743525	0.0010688	0.1889850	0.0010592
Q2	0.1146919	0.0037127	0.1088151	0.0020179
Q3	0.1090127	0.0013898	0.1257802	0.0016618
Q4	0.1462186	0.0028428	0.1043107	0.0018131
Q5	0.1391565	0.0022554	0.2379355	0.0111596
Q6	0.1246219	0.0016452	0.2268318	0.0025159
Q7	0.1020386	0.0014686	0.1680096	0.0019838
Q8	0.1099152	0.0011290	0.1914059	0.0010930
Q9	0.0705571	0.0012097	0.0971323	0.0011827
Q10	0.0900616	0.0011907	0.1091233	0.0010717
Q11	0.1212053	0.0011059	0.1911143	0.0010356
Q12	0.1332350	0.0012631	0.2111791	0.0013082
Q13	0.1217054	0.0013704	0.2193553	0.0017975
Q14	0.1300057	0.0063233	0.1944020	0.0012449
Q15	0.0853827	0.0016110	0.1256537	0.0019941
Q16	0.0746527	0.0011631	0.1303570	0.0014266
Q17	0.1011552	0.0011564	0.1737531	0.0010264

The response time of query execution in the phases P3 and P4 of standalone system are presented in Table 3.

The results in Table 2 and Table 3 show that response time of query execution for all queries with caching is lower than without caching. The improvement is significant with all types of queries. The bigger the involved data the higher improvement is resulted by applying caching.

Table-3. The result of P3 and P4 in standalone system

	P3		P4	
Query	Non-Cache	Cache	Non-Cache	Cache
Q1	0.5396976	0.0010229	2.2130160	0.0010281
Q2	0.6830338	0.0262220	3.5772230	0.1553261
Q3	0.4846241	0.0043907	1.8988990	0.0130687
Q4	0.7431542	0.0171590	4.1549310	0.1081332
Q5	0.7401709	0.0114675	3.0402140	0.0022376
Q6	0.6032453	0.0068385	2.2505520	0.0278870
Q7	0.5096860	0.0045215	2.2431590	0.0176957
Q8	0.6768925	0.0009779	2.3414040	0.0011835
Q9	0.4644715	0.0017498	1.8870290	0.0026999
Q10	0.4263281	0.0009145	1.9058490	0.0010663
Q11	0.6661291	0.0010140	2.4589780	0.0010838
Q12	1.1911524	0.0011866	2.9607830	0.0012625
Q13	0.7462513	0.0013488	2.4844530	0.0012071
Q14	0.7089699	0.0010861	2.4446860	0.0011466
Q15	0.4547726	0.0032481	1.9272760	0.0124869
Q16	0.4815077	0.0015633	1.9333240	0.0031702
Q17	0.5305893	0.0010612	2.1488340	0.0011355

Table 4 and Table 5 present the results for client-server system.

Table-4. The result of P1 and P2 in client-server system

	P 1		P2	
Query	Non-Cache	Cache	Non-Cache	Cache
Q1	0.0019607	0.0338124	0.0018267	0.0673393
Q2	0.0058976	0.0078376	0.0045933	0.0149182
Q3	0.0024565	0.0035589	0.0040213	0.0068385
Q4	0.0056106	0.0076440	0.0042841	0.0055331
Q5	0.0045032	0.0678604	0.0155843	0.1572880
Q6	0.0041579	0.0397551	0.0057334	0.0764448
Q7	0.0025960	0.0375549	0.0039457	0.0784872
Q8	0.0024780	0.0449281	0.0026435	0.0945814
Q9	0.0019519	0.0028731	0.0028062	0.0051634
Q10	0.0020485	0.0032194	0.0028013	0.0033805
Q11	0.0021062	0.0537042	0.0021254	0.1177131
Q12	0.0022890	0.0651954	0.0027685	0.1390019
Q13	0.0027707	0.0586838	0.0042931	0.1213539
Q14	0.0019491	0.0515476	0.0039212	0.0051362
Q15	0.0025961	0.0042301	0.0022024	0.1126135
Q16	0.0021842	0.0030673	0.0025272	0.0045962
Q17	0.0021624	0.0334407	0.0030330	0.0700180

Table-5. The result of P3 and P4 in client-server system

	P3		P4	
Query	Non-Cache	Cache	Non-Cache	Cache
Q1	0.0018960	0.6601156	0.0435279	4.1322056
Q2	0.0280391	0.6449867	0.3318795	4.4872204
Q3	0.0137948	0.0353426	0.0160832	1.6769764
Q4	0.0203815	0.5425361	0.1821219	4.9271134
Q5	0.0140861	0.8600350	0.0042396	5.2070280
Q6	0.0107668	0.6401707	0.0387153	7.2895811
Q7	0.0066079	0.6231424	0.0222229	5.2179774
Q8	0.0021964	0.7697019	0.0019169	4.8541472
Q9	0.0030772	0.0146861	0.0049721	1.9258970
Q10	0.0021125	0.0180916	0.0020366	1.9683380
Q11	0.0020450	0.8815631	0.0020068	5.6550891
Q12	0.0024103	1.0496134	0.0023406	6.2858812
Q13	0.0021318	0.9181136	0.0020022	5.9463682
Q14	0.0023554	0.5764136	0.0020882	5.6825592
Q15	0.0054374	0.0316479	0.0150830	1.5791501
Q16	0.0028682	0.0162668	0.0052960	1.9990924
Q17	0.0020640	0.7048124	0.0020359	3.7366625

The experiments in client-server system indicate the same results. The response time of query execution for all queries with caching is lower than without caching. The results with bigger data (P4) indicate that caching produce bigger saving. In this case, saving is the ratio of the response time of the system with caching and without caching. For example, saving for Q1 with P1 in client-server system is 94.20% ((1 - (0.0019607 / 0.0338124)) * 100). The average saving for all queries in both environment is presented in Table 6.

Table-6. The average saving

Environment	Average Saving (%)			
	P1	P2	P3	P4
Standalone	98.31	98.77	99.21	99.35
Client-server	68.26	71.93	92.89	99.06

Table 6 shows that the saving for all phases are mostly more than 90%, this indicates that the implementation of caching can reduce response time significantly. The highest improvement is achieved when it is applied into big size of data in client-server system

A statistical analysis is conducted to find out the difference of the two studied methods. The response time of each query in all phases is compared by using Wilcoxon Signed Test. This non-parametric test is chosen since the number of sample data is small (17). The p-values resulted from all comparison (all phases in both environment are less than 0.05. It indicates that the two methods are significantly different.

CONCLUSIONS AND SUGGESTIONS

The implementation of query cache into a database has been investigated experimentally in this research. The response time of query execution in database with caching is compared with database without caching. The two methods are implemented and compared in two environments, standalone and client-server system. The results show that the application of caching reduce the response time of query execution with big saving. The results also indicate that the application of caching in bigger data will produce bigger benefit. It has been analyzed statistically that the query caching can significantly improve the response time of a database.

However, the validity of this experiment needs to be enhanced in the next research. The application and data under tests with real operation need to be applied. It is also suggested to investigate the methods in more complex database with varies of query.

REFERENCES

Alexandros L., Qiong L. Jie X., and Wenwei X. 2010. Caching and Materialization for Web Databases. *Found. Trends databases* 2, 3 (March 2010), pp. 169-266.

Charles G., Amit M., Anastasia A., Bruce M., Todd M., Christopher O., and Anthony T. 2008. Scalable query result caching for web applications. *Proceedings Very Large Data Base* Endowment. 1, 1 (August 2008), pp. 550-561.

Immanuel C. and Lance A., "Oracle® Database Performance Tuning Guide 11gRelease 2 (11.2)", Oracle, 2014

Jesse A., Lawrence J., Xiang L., Jordan P., Zheng Z., Tie Z., Web caching for database applications with Oracle Web Cache, *Proceedings of the 2002 ACM SIGMOD international conference on Management of data,* June 03-06, 2002.

Larson, P.-A., Goldstein, J., Jingren Z. "MTCache: transparent mid-tier database caching in SQL server", *Data Engineering, 2004. Proceedings. 20th International Conference on,* pp. 177 – 188.

Mehmet A., Qiong L., Sailesh K., C. Mohan, Hamid P., Bruce G. L., Honguk W., Larry B., DBCache: database caching for web application servers, *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, June 03-06, 2002.

Paweł L. and Krzysztof S. 2010. Consistent caching of data objects in database driven websites. In *Proceedings of the 14th east European conference on Advances in databases and information systems* (ADBIS'10), Barbara Catania, Mirjana Ivanović, and Bernhard Thalheim (Eds.). Springer-Verlag, Berlin, Heidelberg, pp 363-377.

Priya G., Nickolai Z., Samuel M., "A Trigger-Based Middleware Cache for ORMs", Middleware 2011, Lecture Notes in Computer Science-Springer 7049, pp 329-349.

Qiong L., Sailesh K., C.Mohan, Honguk W., Hamid P., Bruce G. L., Jeffrey F. N., "Middle-tier database caching for e-Business", *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, Madison, WI, June, 2002.

Qiong Luo, Jerey F. Naughton, Rajasekar Krishnamurthy, Pei Cao, and Yunrui Li, "Active Query Caching for Database Web Servers", *International Workshop on the Web and Databases-WebDB 2000*, Lecture Notes in Computer Science-Springer 1997, pp. 92-104, 2001.

Schwartz B. et al.: "High Performance MySQL 3rd Edition", O'Reilly Media, United States of America, 2012

http://www.merriam-webster.com/dictionary/cache Merriam-Webster online dictionary. Accessed on26 April 2014

http://dev.mysql.com/doc/refman/5.5/en/query-cache.html. Accessed on 15 April 2014

http://dev.mysql.com/doc/refman/5.5/en/query-cache-operation.html. Accessed on 15 April 2014

http://dev.mysql.com/doc/refman/5.5/en/query-cache-in-select.html. Accessed on 15 April 2014