Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2017) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

CENTERIS - International Conference on ENTERprise Information Systems
CENTERIS 2017, 8-10 November 2017, Barcelona, Spain

Identifying New Directions in Database Performance Tuning

[a]Derek Colley[*], [b]Dr. Clare Stanier

[a]*Staffordshire University, College Rd., Stoke on Trent, ST4 2DE, United Kingdom*
[b]*Staffordshire University, College Rd., Stoke on Trent, ST4 2DE, United Kingdom*

Abstract

Database performance tuning is a complex and varied active research topic. With enterprise relational database management systems still reliant on the same set-based relational concepts that defined early data management products, the disparity between the object-oriented application development model and the object-relational database model, called the object-relational impedance mismatch problem, is addressed by techniques such as object-relational mapping (ORM). However, this has resulted in generally poor query performance for SQL developed by object applications and an irregular fit with cost-based optimisation algorithms, and leads to questions about ~~the~~ need for the relational model to better adapt to ORM-generated queries. This paper discusses database performance optimisation developments and seeks to demonstrate that current database performance tuning approaches need re-examination. Proposals for further work include exploring concepts such as dynamic schema redefinition; query analysis and optimisation modelling driven by machine learning; and augmentation or replacement of the cost-based optimiser model.

Peer-review under responsibility of the scientific committee of the CENTERIS - International Conference on ENTERprise Information Systems / ProjMAN - International Conference on Project MANagement / HCist - International Conference on Health and Social Care Information Systems and Technologies.

Keywords: Database, SQL, performance tuning, cost-based optimiser, object-relational mapping, object-relational impedance mismatch

* Corresponding author. Tel.: +44-7745-467984
E-mail address: derek.colley@research.staffs.ac.uk

Peer-review under responsibility of the scientific committee of the CENTERIS - International Conference on ENTERprise Information Systems / ProjMAN - International Conference on Project MANagement / HCist - International Conference on Health and Social Care Information Systems and Technologies.

1. Introduction

Relational database systems (RDBMS) underpin a large number of today's business enterprise applications, ranging from payment processing systems for e-commerce websites to large-scale Customer Relationship Management (CRM) systems. Based on the seminal work of Codd [14], RDBMS have been in use for almost half a century and RDBMS performance tuning is a well-understood field. Relational databases have been extended over time to include object-oriented support and integration with external languages [35] and as application development paradigms have advanced, performance issues have emerged. This is particularly the case when dealing with non-static, fluctuating application models which interface with RDBMSs through paradigms such as entity-framework modelling. Automatic generation of SQL can lead to sub-optimal query performance, and one response to performance issues has been the emergence of NoSQL databases. However, relational databases remain the most widely used database technology, especially in traditional business environments, and there is a need to identify new approaches for performance tuning to keep pace with the progress in application development methodologies. This paper reviews existing RDBMS performance optimisation methods and techniques to identify the strengths and limitations of traditional approaches in the current database environment and suggests directions for future work. The rest of this paper is organised as follows. Section 2 gives the context of the investigation. Section 3 discusses current approaches to database performance tuning and Section 4 gives the conclusions and suggestions for future work.

2 Research Approach

A systematic literature review of RDBMS performance tuning approaches was conducted. Stage 1 was to seed and search the literature, identifying seminal papers through key phrase searches. There is an extensive literature on relational performance tuning and for this reason, Stage 2 ranked the papers using a citation function to prioritise key sources. Stage 3 was to analyse the resulting sources, extracting the topics and key conclusions, and fitting them into a directed graph. The process was then iterated, using the results from Stage 3. More than 100 highly cited papers were identified but the volume of material relating to RDBMS performance tuning means that only the sources identified as most relevant are discussed in this paper. The process is shown in Fig. 1.

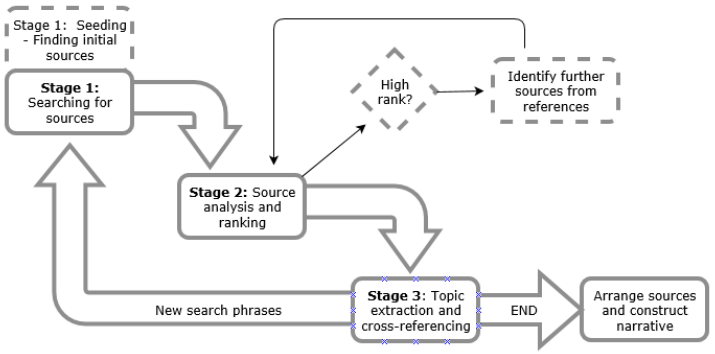

Figure 1: Source Identification Strategy

A memo-ing approach adopted from grounded theory was used to structure the research and support the identification of themes. This approach meant that a large amount of data could be identified, analysed, categorised into a taxonomy, linked and summarised quickly, while examination of each source yielded valuable information in the form of related topics and general learning. A limitation of the approach was that the method is retrospective. For this reason, the review was expanded to ensure that more recent research was also included.

3 Performance Tuning

From the literature review, three key areas relating to query performance were identified, database design, query optimisation and query design.

3.1 Database Design Considerations

Relational databases are based on relational set theory and effective RDBMS design supports queries based on the relational algebra. From the starting point of Codd [14], extensions and revisions to the model were frequent; Google Scholar lists over 6,400 individual published academic papers and books containing the keywords 'relational database' from 1980 to 1985 alone. However, although relational design concepts are well understood, adherence to good database design patterns is not enforced in the industry, nor arguably is it now even encouraged [1]. The primary technique for achieving optimal relational design is normalisation [15, 16], although normalisation is often criticised for unnecessary complexity [8]. Westland [44] cites the inefficiencies caused by the application of normalisation - one instance is that the number of tables the query must reference causes an increase in JOIN operations, complexity of filters, and consequently an overall increase in query execution cost. Optimising JOIN performance is a continuing theme in the relational database literature ([4], [5], [9], [10], [30], [31], [46]). A key assumption in the work on JOIN optimisation is that query design is driven by the schema design; in other words, that queries are developed to work with a given schema as efficiently as possible. This is not necessarily the case with SQL queries generated by applications or through mapping. Techniques such as entity-framework modelling are not well-suited to creating queries that run efficiently against heavily-normalised or complex schemata.

3.2 Query Execution Optimisation

Indexing is used to reduce the computational and I/O subsystem loads when fetching data. The most widely used type of index is the B/B+ tree index [19] although other types of indexes are common [40]. The limitations of indexes include performance penalties on write-heavy tables [17] and the overhead of indexes themselves [38]. Storing redundant indexes can lead to unnecessary overheads associated with maintenance and storage. Poor query design can mean that the RDBMS engine cannot apply indexes accurately, meaning that indexing can become inefficient. Other strategies to improve relational performance include partitioning ([26]); load balancing ([3]); and varying transaction isolation levels ([20], [29]). As with performance tuning based on efficient design, the underpinning assumption is that optimisation strategies implemented at database level will be used in queries developed at application level and that the query design is based on an understanding of relational optimisation techniques, which as already noted is not necessarily the case for queries automatically generated by object-relational mapping.

Four steps are typically involved in query optimisation in RDBMSs [39]. The cost-based optimizer (CBO) is the current approach for the optimization step, replacing rule-based optimisers which are less efficient when dealing with complex queries ([12]). It is argued here that we are now also reaching the limits of the cost-based optimiser since application and framework generated queries are not necessarily generated in a format which means they fit the CBO approach. Cost-based optimisation techniques share the concept of attempting to reduce the cost of a query (measured by a variety of factors including time taken to execute; page accesses; selectivity factors; cardinality estimates; data density and more). The object is that the least costly plan is chosen ([2]). In most implementations, exhaustive searches are not undertaken due to the cost of calculation; instead, heuristics or timeout parameters are used as a stop condition to select a suitable query plan ([20], [34]).

One limitation of the CBO is that cardinality is a principal factor in calculating the costs of a plan, since the number of rows is normally in direct proportion to the disk accesses required or the size of the dataset returned. Single-dimensional statistical histograms enable reasonably accurate cardinality estimations given predicates ([33]) However, when multiple attributes are involved in a query this attribute-value independence (AVI) becomes a problem since the cardinality error multiplies proportionally to the number of attributes involved ([18]) and the intermediate relations ([13]). As early as 1988, Kabra and DeWitt [22] proposed generating statistics on intermediate result sets in the execution plan and argued that query optimisers were not suited to deal with RDBMS platforms that incorporate object-oriented features. It can also be argued that the CBO is not suited to queries from intermediate object-relational mapping tools that are not optimised for the relational model. Wu et al [45] investigated whether cost-based optimiser models were now unusable. Another issue is where CBOs are not able to fully assess the query, as with highly nested queries, the likelihood of choosing the best plan is lowered.

Trummer and Koch [42] examined a "multi-objective parametric query optimization" approach - replacing costs with a function that computes a score from various inputs such as selectivity, predicted execution time or complexity. This fits into the CBO model, but arguably shows a change of direction towards more intelligent means of optimisation and lends weight to the argument that the relational model and the methods for managing it must evolve to meet the expectations of a more dynamic application-led environment.

3.3 SQL Query Design

SQL has been described as an "elephant on clay feet" ([1]) but SQL syntax is relatively straightforward, despite the expansion of the standard. When the RDBMS encounters a JOIN, it can classify and execute the JOIN in a number of different ways; hash JOINs ([4], [5], [10], [46]); the nested-loop JOIN ([31]); the sort-merge JOIN ([9], [21], [30]). Improving JOIN performance is a current research area ([3], [26]). Poor JOIN performance can result from many causes including over-normalisation ([20]); data skew, ([24]); or external factors such as network performance ([36]) and processor architecture ([23]). Other related research topics include sort; tuning aggregations ([6], [7]); the role of views ([27]); cache management ([11], [43]) using set-based logic over loop-based logic ([37]); optimising for OLAP ([32]); parallelism in query execution ([25]); and typing data ([41]).

Queries generated by entity-framework modelling tools can produce queries with multiple levels of nesting and large numbers of base tables, increasing the number of relations from which to extract data and increasing the query execution load through additional operations on the data (filtering and sorting). It has also been recognised that poor execution performance may result from automatically-generated SQL syntax produced by an entity-framework modeller ([28]).

4 Conclusions and Future Work

Since the inception of the relational model, there have been few significant changes to fundamental concepts. The growth of object-oriented application development in the 1990s led to the demand for support for object-oriented components in relational systems and the transition to object-relational systems delivered in the form of programmatic elements such as stored procedures and user-defined types, materialised views, and extensions to SQL. The demand for better data storage systems led to the incorporation of object-oriented concepts such as R/Python data analytics, XML and JSON support, OLAP integration, and user-defined functions. However, the development of entity-framework models has led to a poor fit between the object-oriented query production process and the query-optimised relational database design paradigm. The ubiquity of relational database systems produced a comprehensive set of strategies and techniques to optimise query performance, but, as the discussion in this paper shows, these strategies and techniques are database-oriented in that they rely for their effectiveness on queries being designed to fit the

database, for example, by structuring the query to be accessible to the CBO. The rise of model-driven queries presents new challenges for relational query optimisation, identifying the need for revisions to the CBO and novel approaches, such as dynamic schema redefinition, or augmentation of the CBO with machine learning techniques, in response to changing inbound query patterns. It is intended these approaches will be the underpinnings of our future research in this area, and these new approaches in the relational space would improve relational query performance for model-generated queries and further address the object-relational impedance mismatch problem.

References

1. Atzeni, P., Jensen, C.S., Orsi, G., Ram, S., Tanca, L. and Torlone, R., 2013. The relational model is dead, SQL is dead, and I don't feel so good myself. ACM SIGMOD Record, 42(2), pp.64-68.
2. Babcock, B. and Chaudhuri, S., 2005, June. Towards a robust query optimizer: a principled and practical approach. In Proceedings of the 2005 ACM SIGMOD international conference on Management of data (pp. 119-130). ACM.
3. Barthels, C., Müller, I., Schneider, T., Alonso, G. and Hoefler, T., 2017. Distributed Join Algorithms on Thousands of Cores. Proceedings of the VLDB Endowment, 10(5).
4. Begley, S., He, Z. and Chen, Y.P.P., 2016. PaMeCo join: A parallel main memory compact hash join. Information Systems, 58, pp.105-125.
5. Blanas, S., Li, Y. and Patel, J.M., 2011, June. Design and evaluation of main memory hash join algorithms for multi-core CPUs. In Proceedings of the 2011 ACM SIGMOD International Conference on Management of data (pp. 37-48). ACM.
6. Borodin, A., Kiselev, Y., Mirvoda, S. and Porshnev, S., 2016, November. Development of data aggregation capabilities in domain-specific query language for metallurgy. In Dynamics of Systems, Mechanisms and Machines (Dynamics), 2016 (pp. 1-6). IEEE.
7. Bose, A., Smadi, M.M., Sun, J. and Velpuri, C.K., International Business Machines Corporation, 2016. Dynamic data aggregation from a plurality of data sources. U.S. Patent 9,292,575.
8. Buelow, R. "The Folklore of Normalization." Journal of Database Management, vol. 11, no. 3, 2000, p. 37.
9. Chen, M. and Zhong, Z., 2014, December. Block Nested Join and Sort Merge Join Algorithms: An Empirical Evaluation. In International Conference on Advanced Data Mining and Applications (pp. 705-715). Springer International Publishing.
10. Chen, S., Ailamaki, A., Gibbons, P.B. and Mowry, T.C., 2007. Improving hash Join performance through prefetching. ACM Transactions on Database Systems (TODS), 32(3), p.17.
11. Chen, T.H., Shang, W., Hassan, A.E., Nasser, M. and Flora, P., 2016, November. CacheOptimizer: Helping developers configure caching frameworks for Hibernate-based database-centric web applications. In Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering (pp. 666-677). ACM.
12. Cherniack, M. and Zdonik, S., 1998, June. Changing the rules: Transformations for rule-based optimizers. In ACM SIGMOD Record (Vol. 27, No. 2, pp. 61-72). ACM.
13. Christodoulakis, Stavros. "Implications of certain assumptions in database performance evaluation." ACM Transactions on Database Systems (TODS) 9.2 (1984): 163-186.
14. Codd, E.F., 1970. A relational model of data for large shared data banks. *Communications of the ACM*, *13*(6), pp.377-387.
15. Codd, E. F. "Recent Investigations into Relational Data Base Systems". IBM Research Report RJ 1385 (April 23, 1974). Republished in Proc. 1974 Congress (Stockholm, Sweden, 1974). , N.Y.: North-Holland (1974).
16. Codd, E.F. "Further Normalization of the Data Base Relational Model". (Presented at Courant Computer Science Symposia Series 6, "Data Base Systems", New York City, May 24–25, 1971.) IBM Research Report RJ909 (August 31, 1971). Republished in Randall J. Rustin (ed.), Data Base Systems: Courant Computer Science Symposia Series 6. Prentice-Hall, 1972.
17. Davidson, L., Ford, T. and Berry, G., 2010. Performance Tuning Using SQL Server Dynamic Management Views. Simple Talk Pub.
18. Faloutsos, C. and Kamel, I. Relaxing the uniformity and independence assumptions using the concept of fractal dimensions. Journal of Computer and System Sciences, 55(2):229–240, 1997.
19. Foster, E.C. and Godbole, S., 2016. Review of Trees. In Database Systems (pp. 471-504). Apress.
20. Fritchey, G. and Dam, S., 2013. SQL Server 2012 Query Performance Tuning. Apress.
21. Graefe, G., 1994, February. Sort-merge-Join: An idea whose time has (h) passed?. In Data Engineering, 1994. Proceedings. 10th International Conference (pp. 406-417). IEEE.
22. Kabra, N. and DeWitt, D.J., 1998, June. Efficient mid-query re-optimization of sub-optimal query execution plans. In ACM SIGMOD Record (Vol. 27, No. 2, pp. 106-117). ACM.
23. Kim, C., Kaldewey, T., Lee, V.W., Sedlar, E., Nguyen, A.D., Satish, N., Chhugani, J., Di Blas, A. and Dubey, P., 2009. Sort vs. Hash revisited: fast Join implementation on modern multi-core CPUs. Proceedings of the VLDB Endowment, 2(2), pp.1378-

1389.
24. Lakshmi, M.S. and Yu, P.S., 2000, January. Effect of skew on Join performance in parallel architectures. In Proceedings of the first international symposium on Databases in parallel and distributed systems (pp. 107-120). IEEE Computer Society Press.
25. Leis, V., Boncz, P., Kemper, A. and Neumann, T., 2014, June. Morsel-driven parallelism: a NUMA-aware query evaluation framework for the many-core age. In Proceedings of the 2014 ACM SIGMOD international conference on Management of data (pp. 743-754). ACM.
26. Lu, Y., Shanbhag, A., Jindal, A. and Madden, S., 2017. AdaptDB: adaptive partitioning for distributed Joins. Proceedings of the VLDB Endowment, 10(5), pp.589-600.
27. Masunaga, Y., 2017, January. An intention-based approach to the updatability of views in relational databases. In Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication (p. 13). ACM.
28. Microsoft Corporation. 2015. Performance Considerations (Entity Framework). [ONLINE] Available at: https://msdn.microsoft.com/en-us/library/cc853327(v=vs.110).aspx. [Accessed 29 March 2017].
29. Microsoft Corporation. 2017. Advanced Query Tuning Concepts. [ONLINE] Available at: https://technet.microsoft.com/en-us/library/ms191426(v=sql.105).aspx. [Accessed 29 March 2017].
30. Mirzadeh, N., Koçberber, Y.O., Falsafi, B. and Grot, B., 2015. Sort vs. hash Join revisited for near-memory execution. In 5th Workshop on Architectures and Systems for Big Data (ASBD 2015) (No. EPFL-TALK-209111).
31. Mishra, P. and Eich, M.H., 1992. Join processing in relational databases. ACM Computing Surveys (CSUR), 24(1), pp.63-113.
32. Myalapalli, V.K. and Dussa, K., 2015, December. Optimizing SQL queries in OLAP database systems. In Information Processing (ICIP), 2015 International Conference on (pp. 833-838). IEEE.
33. Oommen, B.J. and Thiyagarajah, M., 2005. Method of generating attribute cardinality maps. U.S. Patent 6,865,567. [ONLINE] Available at: https://www.google.com/patents/US6865567 [Accessed 01 March 2017]
34. Oracle Corporation (2017) Oracle Database Performance Method. [ONLINE] Available at: https://docs.oracle.com/cd/E11882_01/server.112/e10822/tdppt_method.htm#TDPPT006 [Accessed: 01/03/2017].
35. Pane, A., Goldy, N., Madoery, F., Kira, E., Reynares, E. and Caliusco, L., 2017. From Relational to a Column-based Database: A quasi-experiment. Revista Eletrônica Argentina-Brasil de Tecnologias da Informação e da Comunicação, 1(6).
36. Perrizo, W., Ram, P. and Wenberg, D., 1994. Distributed Join processing performance evaluation. In 1994 Proceedings of the Twenty-Seventh Hawaii International Conference on System Sciences.
37. Ram, S. and Khatri, V., 2005. A comprehensive framework for modeling set-based business rules during conceptual database design. Information Systems, 30(2), pp.89-118.
38. Randal, P. 2015. On index key size, index depth and performance. [ONLINE] Available at: https://www.sqlskills.com/blogs/paul/on-index-key-size-index-depth-and-performance/. [Accessed 28 March 2017].
39. Selinger, P.G., Astrahan, M.M., Chamberlin, D.D., Lorie, R.A. and Price, T.G., 1979, May. Access path selection in a relational database management system. In Proceedings of the 1979 ACM SIGMOD international conference on Management of data (pp. 23-34). ACM.
40. Shahvarani, A. and Jacobsen, H.A., 2016, June. A hybrid b+-tree as solution for in-memory indexing on cpu-gpu heterogeneous computing platforms. In Proceedings of the 2016 International Conference on Management of Data (pp. 1523-1538). ACM.
41. Šimenić, K., Lukić, I. and Köhler, M., 2015, January. Geolocation and Data Type Impact on SQL Query Execution Time Using Different Database Approaches. In 38th international convention on information and communication technology, electronics and microelectronics.
42. Trummer, I. and Koch, C., 2017. Multi-objective parametric query optimization. The VLDB Journal—The International Journal on Very Large Data Bases, 26(1), pp.107-124.
43. Welborne, C.R., de Voogt, D. and Eatough, M., 2016. An analysis of database caching policies. Journal of Computing Sciences in Colleges, 32(2), pp.4-10.
44. Westland, J.C., 1992. Economic incentives for database normalization. Information processing & management, 28(5), pp.647-662.
45. Wu, W., Chi, Y., Zhu, S., Tatemura, J., Hacigümüs, H. and Naughton, J.F., 2013, April. Predicting query execution time: Are optimizer cost models really unusable?. In Data Engineering (ICDE), 2013 IEEE 29th International Conference on (pp. 1081-1092). IEEE.
46. Zeller, H. and Gray, J., 1990, August. An Adaptive Hash Join Algorithm for Multiuser Environments. In *VLDB* (Vol. 90, pp. 186-197).