International Journal of Database Theory and Application
Vol.9, No.10 (2016), pp.351-362
http://dx.doi.org/10.14257/ijdta.2016.9.10.30

Dynamic Integration of Pl/Sql for Complex Queries

Muhammad Qasim Memon[1,2], He Jingsha[1,2], Aasma[3], Allah Ditta[4] and Khurram Gulzar Rana[4]

[1]*School of Software Engineering, Beijing University of Technology, 100124, Beijing, China*
[2]*Beijing Engineering Research Center for IoT Software and Systems, Beijing University of Technology, 100124, Beijing, China*
[3]*School of Economics and Management, Beijing University of Technology 100124, Beijing, China*
[4]*School of Computer science, Beijing University of Technology 100124, Beijing, China*
memon_kasim@yahoo.com, jhe@bjut.edu.cn, memon.aasma2@gmail.com, hafizad88@yahoo.com, Khurramrana642@gmail.com

Abstract

Query optimization is a very complex task for commercial databases in case of performance issue that needs to be well known of entire structure of database. In desktop or web application at the back end, query processing aspires to be major factor for finding the better execution. We address the problem of SQL query optimization merely from the perspective of query response time in different databases invoked multiple queries are imparted in admission database management system, leveraging join and complex queries. Our proposed method adhering with respect to the underlying topics, to tune (Select, complex and join SQL) queries with optimized execution plan using PL/SQL features by incorporating database objects such as procedures, triggers and methods to improve query performance, instead amendment of query semantics which lessen time of developer or administrator to do manual tuning.

Keywords*: Query Optimization, SQL Tuning, Query Processing, PL/SQL, Database Tuning*

1. Introduction

The main phase of database system performance is tuning of SQL statements, which consists of three steps: first, to determine top SQL statements, of those are time consuming in execution. Those statements are accountable for the applications workload and system resource, by analyzing past SQL execution history available in the system. Second, execution plan created by the query optimizer for those SQL statements perform fairly. Third, to produce better execution plans by employing disciplinary actions on poorly performed SQL statements [4-5,11]. In this paper the clinical aspect of tuning of database system resembles in a way so that performance decreases by consuming less response time with minimum system resources. These non-subjective tasks can be completed in way that complete workload can be managed and executed well to enhance the performance of the database. On the contrary there are some issues with the data manipulation, which represent the required results achieved by reducing, balancing and paralyzing the workload. To surpass these challenges many of the research groups and commercial databases and tools vendors come to resolves these issues over tuning [20]. In this paper, contributions are to improve query response time through various tuning approaches are implemented using complex queries. Evaluating the difference between

ISSN: 2005-4270 IJDTA

the manual and automatic tuning. Basically there are lot of manual approaches are manifested by the developers and administrators for optimizing the performance of databases, even though currently many databases (Oracle, SQL server) offer tuning for query optimization [22,25], but they serve as an external tool for tuning which needs to be well known of the entire workload, and data administrator has to come with lots of expertise. Their efforts have emphasized on several topics such as some of the tool vendors [1], uses optimizer externally to enhance performance itself by providing key statistics triggered through datasets or to reconstruct the SQL statements into good verse parsing techniques by suggesting indexes appropriately. However, practically for financial and daily bases transactional applications like banking side there are many desktop applications developed through various programming languages, these applications are required to query the database that carries lots of transactions. Structured query language (SQL) used to communicate with databases. In this paper query execution method is applied not by only optimizing simple, join or complex queries but with procedural language/structure query language (PL/SQL) by deploying data objects in form of procedures and triggers entitled as tuning (manual and automatic). Rest of paper contributed as, second section literature that describes some important notes of recent approaches. In the third section database structure briefly explained, workload used in the database and the database objects used in PL/SQL. Fourth section explain proposed types of tuning performed in our methodology in comparison to existing methods offered by commercial databases elaborating the difference of the query response time. And final section imparted with conclusion on the future aspect of proposed work.

2. Related Work

LEO (LEarning Optimizer) finds out the column error selectively in its execution plan which is being exploited in the end of execution results by the optimizer. In conclusion the SQL profiling provides the optimizer in detecting the effective optimized plan by making its plan search algorithm. Rather LEO proposed with optimal plan through its repetitions process of query execution that requires each time the query to be executed by the application tool in an iterative process until the required optimization plan is found. Another comparison in the analysis of LEO's strategy is the usage model. As LEO put all those statements for execution and collect correction for each of the statement where the SQL profile only selects only subset of SQL statements that have highly impact over performances [1]. The developing conflict of applications, large volume of data and enormous data structures with monumental data appearing as disputing cause. Query optimizer is main part of Database Management System (DBMS) that runs transactions with several ways as one of it is cost-effectively. These different ways select better execution plan relevant to the atmosphere and data input sources. Developers tunes database to produce optimal execution plan for a query. Apparently it is being considered to create the database system which contains automatic features. These database systems are being referred to as automatic database system management, deployed to decrease reliability with costly human resource. These automatic features of optimizer are given in many commercial databases nowadays with lot of intention to generate required possible results. This research is emphasized to search and allot those points in query optimizers where the human interruption is needed. Optimizers are identified with their automatic features exploits their pros and cons and produces way of promoting the current position of automatic computing in optimizers. The autonomic manner of optimizer is analyzed by remaking and processing several queries though the real time workload experiments with major suggestions [2,6-8,12]. Learning based models proposed for query execution in [3], restricts the requirement for the workload that must be available for training purpose, the limitation is subjected to transactions on daily basis however, for online transactional processing (OLTP), it's very challenging task as it is impractical for workload to be

trained on daily basis if transactions rollback or any data manipulation language (DML) changes occurs. In the past many researchers have laid their efforts to work on their analysis for the processing of complex and join queries [9-10,16-17] in which proposed cost models for complex query plan for identifying good evaluation strategies so that each database system must be processed parallel in multiprocessing environment. Oracle developed many choices to lift up the performance [14,24] like normalizing tables to avoid redundancy, introducing constraints for restricting the integrity and repetition. In this paper, we focus on those techniques to assist administrators to enhance the capability of optimization in their packaged application source code. In this paper the integrated feature of Oracle 11g highlighted the SQL tuning advisor capability by implementing the recommendation to restructure the SQL structure, access design, and query optimization thus making the better performance of high load SQL statements [18] which takes so much time in execution. In this paper two of strategies are followed to tune the queries or transaction being carried out. One of strategy manipulated the same manual work of tuning as developers used with it and another strategy is driven with the autonomic feature of SQL tuning advisor leaning the most efficient execution plans and query optimization with less system resources and lesser time consumption as compared to the other databases [13].

3. Database Structure

In the aspect of data modeling examines the logical structure of the database that determines the overall work flow from top to bottom and storage issue for database also defines how data is executed as it is very much necessary for the performance factor. Here oracle developer data modeler is used to design the database because it states the overall operations those have to be performed at physical model, this can be lead to a logical model, which is one of the main reason to build the modelling structure for any database to measure its scalability and performance rate such as transactions, cost, data consistency and maintenance. A normalized database according to E.F Codd's rules so that each relation in a table must be normalized. This normalized database contains tables, each table has its prescribed data as table name describes like in student table there are eighteen attributes used and other table depicts over all tables in the Figure 3.1.

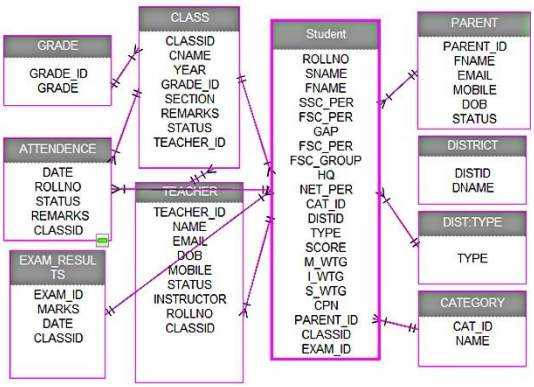


Figure 3.1. Database Schema

4. Query Structure and Execution

In this paper, we have used dataset like in the desktop applications used on daily bases that contains thousands of records. For the workload various records have been added to the system making it real time workload, also provoked thirty queries coded with procedural language (PL) and Structured Query Language (SQL). All these transactions are implemented using different database system such as PostgreSQL, Oracle 11g and SQL server 2008. The need here to differentiate several database vendors to compare the strategy of optimizing queries using data objects as it also reflects performance as database varies with query response time. Among those database systems, best execution plan and cost based query optimization evaluated various results through automatic SQL tuning and manual tuning. In both cases same real work load has been utilized with different tools and techniques of tuning and query execution plan analysis in collaboration with the performance issues. The aim behind this comparison is to compare the response time for each query processing in terms of (Automatic & Manual) SQL tuning and to make sure database optimized in efficient manner. In this proposed framework Procedural language (PL) embedded with PL/SQL code where 80% of the glitches associated with database performance can be fixed by tuning complex SQL queries. We refer the PL/SQL as the process of integrating dynamically PL-SQL Engine environment. Here is the one of the proposed method named as the Add_Std procedure to insert student record as described in Procedure 1 and 2. The database objects describe data manipulation transactions while deploying database objects on insertion of record, this makes query to perform well enough and reduces response time without analyzing queries further by externally providing any recommendations as some of databases or tools offers. As this kind of approach to define database objects embedded sometime in queries or invoking those objects in it, itself is a desired approach here.

Procedure 1. ADD_Std method, describes query embedded in the procedural way that can be invoked from main method.

```
Create or replace procedure ADD_Std
(Assign Attributes and their data types)
  Is
    If
    Parent Record not found
    //Student father exits or not
    Exception (//Parent record not found)
    Begin
      Insert into table_name
      //SQL Query
      If
      Student_Record<>Student_Record
      //Student Exist or Not
      Commit
      Else
      Rollback
      //Student inserted successfully
      If
      Student already exist
      Exception (// record duplication);
END
```


Procedure 2. Main_proc method describes database events as to call any procedure or execute query.

```
Create or replace Main_proc
  BEFORE INSERT OR UPDATE OF Student_table_attribute ON
  Table_Student FOR EACH ROW
  WHEN (condition for student's grading, marking and attendance) BEGIN
  IF INSERTING (condition/ Call procedure)
  Select Table attribute
  From Table; //SQL Query
       END IF;
    END Main proc;
End
```

4.1. Queries

A query is simply way to ask database to perform the operation and retrieve those required records from the table. There are many of types of queries used in this context that may include simple queries, complex queries. Simple queries are those ones that retrieve data from only one table and do not restricts much of the data to be bounded on request. Complex queries are those which retrieve data from more than one table and restrict data to be bounded on certain conditions. In this paper, workload consists of both types of the queries that are focused to tune, but simple queries not enough, hardly two or three queries involved in 30 queries rest of all queries are emphasizing complex queries. Here are some of the queries used randomly among queries:

```
Select rollno, std_name, distinct(distid)
from std s, dist d, dtype dt,cat c
Where s.distid=d.distid
And s.cpn=(select max(s.cpn) from std)
And s.catid=c.catid
And s.type=dt.type
Order by distid

Select * from std s, dist d, dtype dt
Where s.distid=d.distid
And s.type=dt.type And s.type='R';

Select * from std s, Dist d, dtype dt, cat c
Where s.distid=d.distid
And s.catid=c.catid
And s.type=dt.type
Order by distid
```

4.2. Query Processing and Execution

In both case studies two different approaches carried out, in which same workload and transactions are used to measure their estimated execution time and compare both results if varies. For comparisons, both approaches are implemented on same workload while running the queries, these are created to ask the database to fetch the data that is needed on demand or for the purpose of business needs. To evaluate this difference, the SQL Developer tool to make better analysis on the execution plan to make the better execution plan for assisting the result comparison. For this four of the queries from the workload are used in manual tuning also with automatic SQL tuning and both of the tuned queries are

processed with the same tool simultaneously. The execution of queries as shown in Figure 4.2.1 depicts the query execution that displays the records of those students who have valid aggregated CPN. This query executed in 350.25 seconds shown in the status bar at the bottom, normally retrieves 426 rows. With these records the CPN is calculated for each student by the main method mentioned in Procedure 2 rather entering data directly into the table can make any time human error while update or maintaining the database. For this, a procedure is recommended as listed in Procedure 1 and a trigger also recommended by the SQL tuning advisor in the automatic SQL tuning approach in oracle 11g. This query execution consumes 78.688 seconds with manual SQL tuning thus it takes more than one minute to execute this query in the Figure 4.2.2.

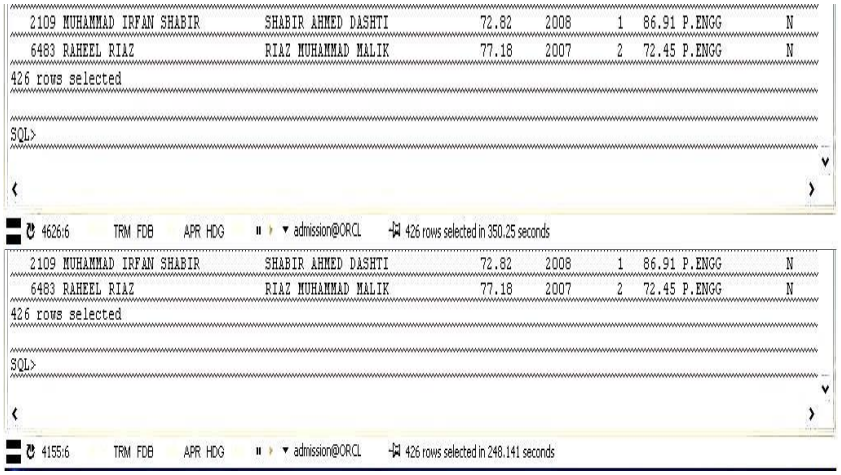


Figure 4.2.1. Query Execution Response Time (Seconds) Oracle 11g (Manual and Automatic SQL Tuning Respectively)

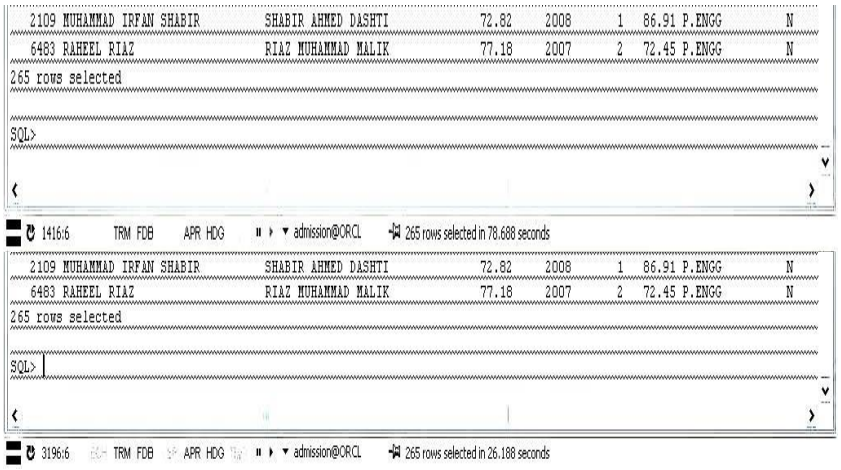


Figure 4.2.2. Query Execution Response Time (Seconds) in SQL Server 2008 (Manual and Automatic SQL Tuning Respectively)

To optimize the query with less time and fewer resources, database is changing rapidly with time, it is needed to maintain the queries execution as the records keep increasing day by day. The query is tuned by automatic tuning optimizer and run through the same tools as executed with the manual tuning and produces the result in just 26.18 seconds as shown in Figure 4.2.3. Complex queries have always issues with performance [14] and these kinds of queries effect so much on the performance that database do not find optimal solution on daily basis transactions. Here the complex queries are taken into consideration and retrieves data from more than one tables based upon the restrictions as

specified. Such as list out students of all districts who are not hafiz e Quran and contains the category of general, special scheme, diploma, employee's quota students as well. Query consist of so many where clauses used to restrict the data to fetch rows are required by query. Response time consume for this query is shown in the Figure 4.2.4. As mentioned in both of the cases with manual tuning, query execution consumes the time approximately 208.32 whereas, automatic SQL tuning with an approximate estimated time consumed 105.06 seconds conceded while query execution.

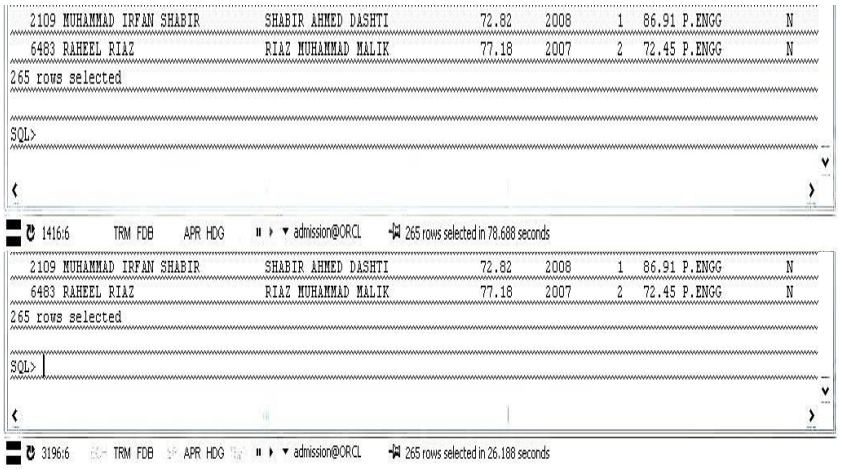


Figure 4.2.3. Query Execution Response Time (Seconds) in SQL Server 2008 (Manual and Automatic SQL Tuning Respectively)

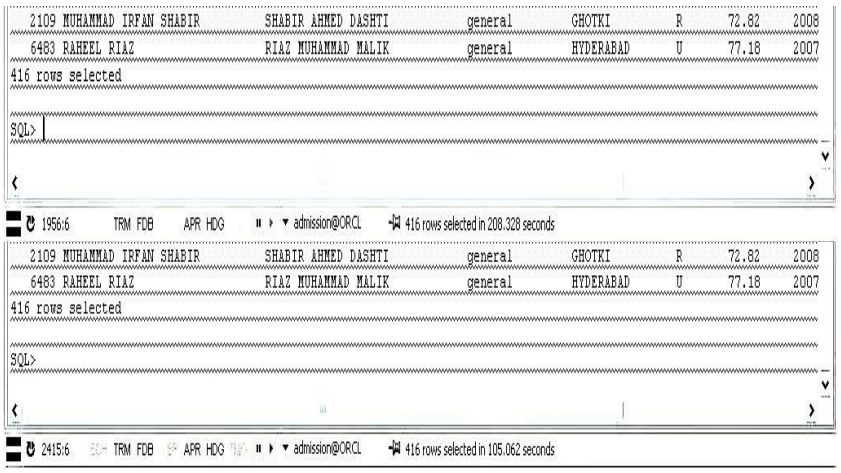
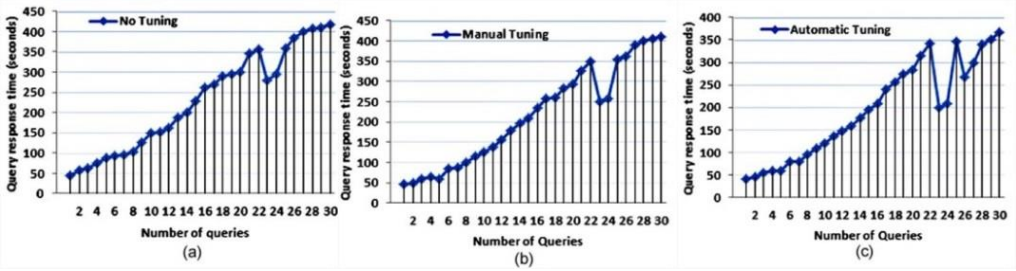


Figure 4.2.4 Query Execution Response Time (Seconds) in Postgresql (Manual and Automatic SQL Tuning Respectively)

In both of the tuning approaches it is pre much mainly the time difference of 107 seconds exempting the process and resources usages. This is quite tremendous performance over manual tuning which can take more time on consuming where thousands of queries execute each day for any organization. This whole process of tuning is administrated by the database developer or the data administrator to reduce the time behind the query execution and it's depends upon the need of the running enterprise as how much they want to maximize their query execution.

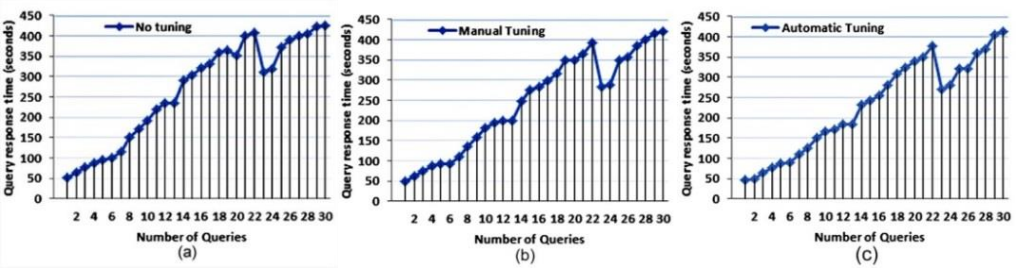
4.3. Experiment Results

We select top thirty queries that are choose to make the cause of performance by using different database management system (DBMS). These queries are executed in three of the tuning approaches these are: without tuning, manual tuning, automatic SQL tuning. The Figure 4.3.1 displays the response time of all the queries and the graph indicates the response time as it gradually increasing after one to another query exempting two queries in the middle. Those queries that take less time to execute are prior in tuning shown by the graph. Figure 4.3.1 depicts query execution, while the response time without tuning is time consuming for the query execution, almost it consumes an aggregated time is 115 minutes to execute of all these queries, with an average of 4 minutes per query in execution, the maximum time for a query approximately 13 minutes, this is so costly for a query to execute in 7 minutes as it falls short of expectation as any one do not need to. All of these queries processed with manual tuning approach that is shown in Figure 4.3.2. After manual tuning the worst time for a query is 6.8 minutes for a query and it produces the cumulative time for all the queries approximately 4 minutes for each of the query.



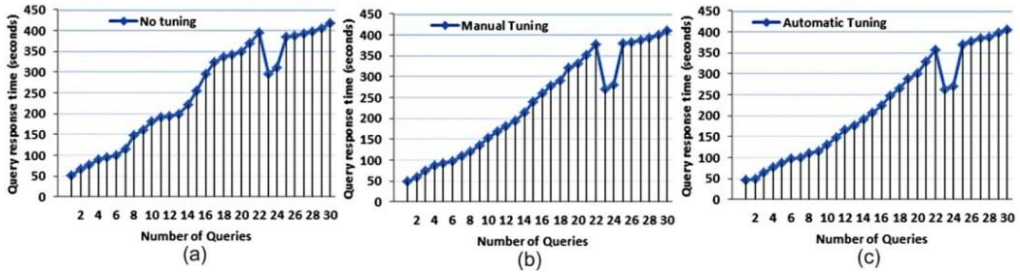
(a) (b) (c)

Figure 4.3.1. Queries Execution Response Time (seconds) in Oracle 11g: (a) No Tuning, (b) Manual Tuning, (c) Automatic Tuning



(a) (b) (c)

Figure 4.3.2. Queries Execution Response Time (seconds) in SQL Server 2008: (a) No Tuning, (Bb Manual Tuning, (c) Automatic Tuning



(a) (b) (c)

Figure 4.3.3. Queries Execution Response Time (seconds) in PostgreSQL: (a) No Tuning, (b) Manual Tuning, (c) Automatic Tuning

This is slightly a good progress in comparison to without tuning but this is still time consuming for any organization to concede 4 minutes for executing each query. This average time can be further minimized with the automatic tuning mode. The time consumed behind each query approximately 3 minutes and the total estimated time for all the queries conceded 97 minutes. Automatic and manual tuning that differs in cumulative response time by 11 minutes as illustrated in Figure 4.3.3. With this approach not only time response is main issue but the resources like processing and input output devices are essential used while query execution. The results are mentioned in a precise way in the Table 1, shows automatic SQL tuning is superior over manual tuning using oracle 11g DBMS in conceding the response time for the better execution and generates the execution plans to achieve the desired goal in case of performance issues. These results describing the manual tuning varies in the aggregated response time, average response time and in maximum response time for the workload. These results state the response time of 30 queries executed by the two approaches of tuning and also without tuning is also incorporated with these results to find out the main difference after tuning between automatic and manual tuning. This is huge difference while optimizing query execution and reducing their estimated time required for the entire workload.

Table 1. Query Execution Response Time (Oracle 11g)

Tuning Mode	Query Response Time in Oracle11g DBMS		
	Maximum	Average	Cumulative
No Tuning	418	229.9	6897
Manual	410	216.7	6501
Automatic	366	195	5850

Table 2. Query Execution Response Time (SQL Server 2008)

Tuning Mode	Query Response Time in SQL Server DBMS		
	Maximum	Average	Cumulative
No Tuning	425	265.3667	7961
Manual	420	246.8667	7406
Automatic	413	231.8	6954

Table 3. Query Execution Response Time (PostgreSQL)

Tuning Mode	Query Response Time in PostgreSQL DBMS		
	Maximum	Average	Cumulative
No Tuning	417	251.4	7542
Manual	411	236.2	7086
Automatic	405	220.8667	6626

To compare the analysis, Figure 4.3.4, 4.3.5 and 4.3.6 illustrates the response time of all queries in SQL server, oracle 11g and PostgreSQL respectively. The main difference in these tuning analyses is that each query performs differently as same workload and transactions used. Response time shows the time to execute each query seconds.

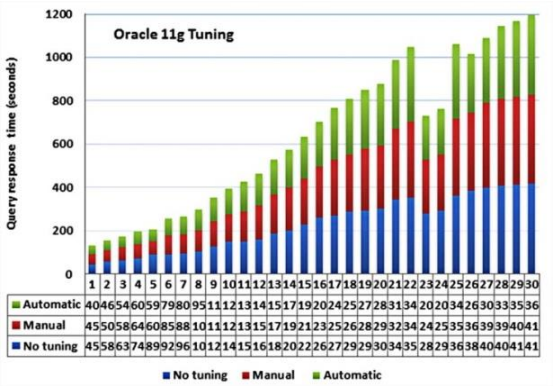


Figure 4.3.4. Oracle 11g Tuning (Automatic, Manual, Without Tuning) Each Query Execution Time is at Bottom for Each Tuning Mode

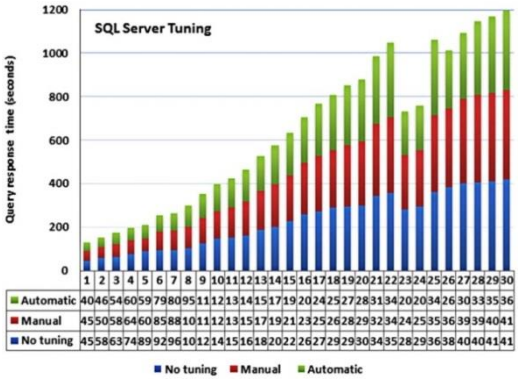


Figure 4.3.5. SQL Server Tuning (Automatic, Manual, without Tuning) Each Query Execution Time is at Bottom for Each Tuning Mode

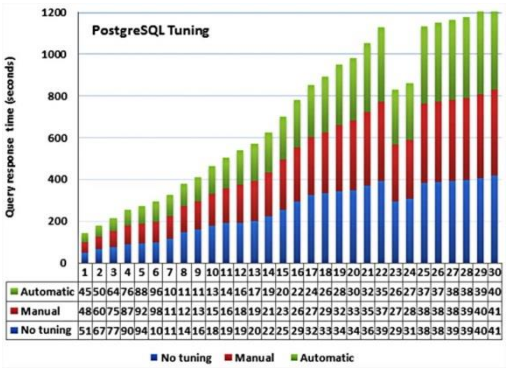


Figure 4.3.6. PostgreSQL Tuning (Automatic, Manual, without Tuning) Each Query Execution Time is at Bottom for Each Tuning Mode

These queries are found by obtaining better execution plan which increase the performance of database, as we have tuned the queries to make sure about cost based execution plan and incorporated with PL/SQL to make better implementation of transactions. In each of the database system implementation the difference in total estimated time varied in each of database as each query takes different response time for execution. Three different database systems showing different response time for each query in each of the tuning mode as Figure 4.3.7 depicts that the automatic tuning in Oracle 11g, with average response time per query is 3.2 minutes if compared to manual tuning it concedes 3.6-minute average response time per query. The cumulative response time difference between manual and automatic tuning is 10.85 minutes. As in the SQL server the average response time per each query is 3.86 with automatic SQL tuning mode and with manual tuning the average response time is 4.1 with cumulative response time difference between both of tuning modes is 7.53 minutes. In PostgreSQL the average response time per query with automatic and manual tuning mode is 3.68 and 3.93 respectively. The difference of cumulative response time for all the queries between both tuning modes is 7.66 minutes.

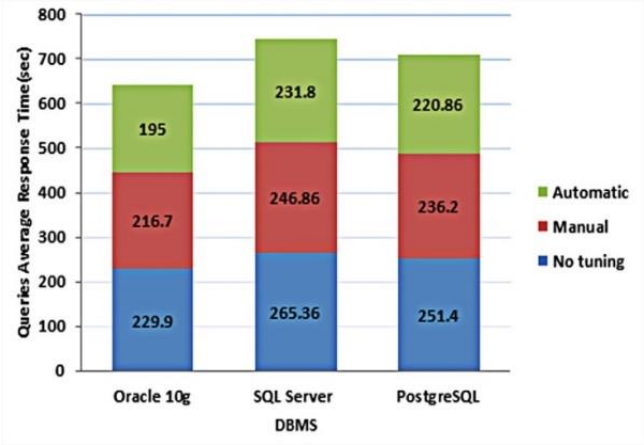


Figure 3.4.7. Average Query Response Time Comparison of Tuning Mode in Each of the DBMS

5. Conclusions

In this paper, to our best knowledge we eradicate all the poor SQL Statements and debug the PL/SQL code making the better execution plan to optimize the queries rather relying only on automatic SQL tuning. We set up the real time workload consists of 30 queries and make amendments by incorporating features of SQL structure and access design. On the first end manual tuning by using the commercial tools, on the other hand end, tuning features merged with automatic SQL tuning. All the results derived from different tuning methods significantly to prevail efficient data retrieval consuming less response time [15,21]. Different database systems used in this paper, Oracle 11g, SQL Server 2008, PostgreSQL has exploited magnificent changes over tuning as workload treated with PL/SQL features, imparted much efficient query response time especially with Oracle 11g tuning. We hope the effort made in this paper can lead to optimize in retrieving data such as cloud environment [23] and crowdsourcing systems [19] with respect to multidimensional data or for decision support system where some of trained classifier used with queries analyzing their response time and other input output resources are subject to research in underlying perspective.

Acknowledgments

The work in this paper has been supported by National Natural Science Foundation of China (61272500), National High-tech R&D Program (863 Program) (2015AA017204) and Beijing Natural Science Foundation (4142008).

References

[1] M. Stillger, G. M. Lohman, V. Markl and M. Kandil, "LEO - DB2 LEarning Optimizer", VLDB, **(2001)**.
[2] R. M. Sher and Hussain, "Autonomic View of Query Optimizers in Database Management Systems", Software Engineering Research, Management and Applications (SERA), Eighth ACIS International Conference, vol. 3, no. 8, **(2010)**, pp. 24-26.
[3] M. Akdere, U. Çetintemel, M. Riondato, E. Upfal and S. B. Zdonik, "Learning-based Query Performance Modeling and Prediction", Data Engineering (ICDE), IEEE 28th International Conference, **(2012)**, pp. 390-401.
[4] H. Herodotou and S. Babu, "Xplus: a SQL-tuning-aware query optimizer", Proceedings of VLDB Endowment, vol. 3, no. 1-2, **(2010)**.
[5] L. J. Muhammad, Y. B. Zakariyau, A. G. Ali and A. Garba, "On Multi Query Optimization Algorithms Problem", International Journal of Database Theory and Application, vol. 7, no. 6, **(2014)**, pp. 13-20.
[6] M. Khan and M. N. A. Khan, "Exploring Query Optimization Techniques in Relational Databases", International Journal of Database Theory & Application, vol. 6, no. 3, **(2013)**.
[7] F. Sun and L. Wing, "Paging Query Optimization of Massive Data in Oracle 10g Database", Computer and Information Science and Service System (CSSS), IEEE International Conference, **(2011)**.
[8] H. Herodotou, N. Borisov and S. Babu, "Query Optimization Techniques for Partitioned Tables", ACM SIGMOD International Conference on Management of data, **(2011)**.
[9] W. Wu, Y. Chi, H. Hacigumus and J. F. Naughton, "Towards predicting query execution time for concurrent and dynamic database workloads", PVLDB, vol. 6, no. 10, **(2013)**, pp. 925-936.
[10] F. Liu and S. Blanas, "Forecasting the Cost of Processing Multi-Join Queries via Hashing for Main-memory Databases", Proceedings of the Sixth ACM Symposium on Cloud Computing, **(2015)**, pp. 153-166.
[11] F. A. C. A. Gonçalves, F. G. Guimarães and M. J. F. Souzab, "Query join ordering optimization with evolutionary multi-agent systems", Expert Systems with Applications, vol. 41, no. 15, **(2014)**, pp. 6934-6944.
[12] R. V. Nehme, K. Works, C. B. Lei, E. A. Rundensteiner and E. Bertino, "Multi-route query processing and optimization", Journal of Computer and System Sciences, **(2012)**, pp. 312-329.
[13] P. Yuan, C. Xie, H. Jin, L. Liu, G. Yang and X. Shi, "Dynamic and fast processing of queries on large-scale RDF data", Knowledge Information System, vol. 41, **(2014)**, pp. 311-334.
[14] D. Kossmann and K. Stocker, "Iterative Dynamic Programming: A New Class of Query Optimization Algorithms", ACM Transactions on Database Systems, vol. 25, no. 1, **(2000)**, pp. 43-82.
[15] A. Yarygina and B. Novikov, "Optimizing resource allocation for approximate real-time query processing", Journal of Computer Science and Information Systems, vol. 11, no. 1, **(2013)**, pp. 69-88.

[16] H. Wang, X. Qin, X. Zhou, F. Li, Z. Qin, Q. Zhu and S. Wang, "Efficient query processing framework for big data warehouse: an almost join-free approach", Frontier of Computer Science, vol. 9, no. 2, **(2015)**, pp. 224-236.
[17] H. Gunasekaran and T. K. Gowder, "New bucket join algorithm for faster join query results", The International Arab Journal of Information Technology, vol. 12, no. 6, **(2015)**.
[18] C. Li, B. He, N. Yan and M. A. Safiullah, "Set Predicates in SQL: Enabling Set-Level Comparisons for Dynamically Formed Groups", IEEE transactions on Knowledge and Data Engineering, vol. 26, no. 2, **(2014)**, pp. 438-451.
[19] J. Fan, M. Zhang, S. Kok, M. Lu and B. C. Ooi, "CrowdOp: Query Optimization for Declarative Crowdsourcing systems", IEEE transactions on Knowledge and Data Engineering, vol. 27, no. 8, **(2015)**, pp. 2078-2091.
[20] A. Cali, D. Calvanese and D. Martinenghi, "Dynamic Query Optimization under Access Limitations and Dependencies", Journal of Universal Computer Science, vol. 15, no. 1, **(2009)**, pp. 33-62.
[21] M. O. Nassar, F. A. Mashagba and E. A. Mashagba, "Improving the User Query for the Boolean Model Using Genetic Algorithm", International Journal of Computer Science Issues, vol. 8, no. 5, **(2011)**, pp. 66-70.
[22] Yarygina, "Execution and Optimization Techniques for Approximate queries in heterogeneous systems", Programming and Computer Software, vol. 39, no. 6, **(2013)**, pp. 309-317.
[23] S. Abiteboul, P. Bourhis and V. Vianu, "High Expressive Query Language for Unordered Data Trees", Theory Computing Systems, vol. 56, no. 4, **(2015)**, pp. 1-40.
[24] A. Badia and A. Wanger, "Complex SQL Predicates as Quantifiers", IEEE Transactions on Knowledge and Data Engineering, vol. 26, no.7, **(2014)**, pp. 1617-1630.
[25] S. Bottcher, R. Hartel and D. Wolters, "S2CX: From relational data via SQL/XML to (Un) Compressed XML", Information System, **(2015)**.

Authors

Muhammad Qasim Memon, received his BE degree in Computer System Engineering and ME in Information Technology from Mehran University of Engineering & Technology Jamshoro (MUET) Pakistan in 2010 and 2014 respectively. Currently he **is** PHD scholar at school of software Engineering in Beijing University of Technology. His research interests include Text mining, machine learning and information security.

He Jingsha, received his B.S. degree from Xi'an Jiaotong University in Xi'an, China and his M.S. and Ph.D. degrees from the University of Maryland at College Park in USA. He is currently a professor in the School of Software Engineering at Beijing University of Technology in China. Professor He has published over 170 research papers in scholarly journals and international conferences and has received nearly 30 patents in the United States and in China. His main research interests include information security, network measurement, and wireless ad hoc, mesh and sensor network security.

Aasma, received her BA and MPA degree from University of Sindh, Jamshoro, Pakistan in 2008 and 2012 respectively. She is currently a PHD scholar at school of Economics and Management in Beijing University of Technology. Her research interests include management information system, human Resource management and data Mining.

Allah Ditta, was born in 1988, in Islamic Republic of Pakistan. He received his MSc degree in Information Technology from Quaid-i-Azam University (QAU) Islamabad Pakistan. He is currently a Ph.D. Scholar at College of Computer Science and Technology in Beijing University of Technology, Beijing China. His research interests include Cryptography, Information Security and Steganography Protocols.

Khurram Gulzar Rana, was born in 1979 in Islamic republic of Pakistan. he is a lecturer in Quaid-i-Azam university Islamabad, Pakistan. he received his MSc degree in information technology from Quaid-i-Azam university Islamabad and his MS degree in information security from national university of science and technology Islamabad Pakistan. He is currently a PHD scholar at college of computer science and technology in Beijing university of technology, Beijing china. His research interests include wireless sensor network, ad hoc network, security of wireless and ad hoc networks, cryptography and network security.