World Applied Sciences Journal 30 (Innovation Challenges in Multidiciplinary Research & Practice): 91-99, 2014
ISSN 1818-4952

DOI: 10.5829/idosi.wasj.2014.30.icmrp.14

Database Performance Tuning Methods for Manufacturing Execution System

Nor'azah Md Khushairi, Nurul A.Emran and Mohd Mokhtar Mohd Yusof

Software Engineering Department, Faculty of Information and Communication Technology,
Universiti Teknikal Malaysia Melaka (UTeM), Hang Tuah Jaya, 76100, Melaka, Malaysia

Submitted: Jan 5, 2014; **Accepted:** Feb 24, 2014; **Published:** Mar 12, 2014

Abstract: In manufacturing industry where data are produced and shared every day, data volumes could be large enough for the database performance to become an issue. Manufacturing Execution System (MES) is such a system that cannot tolerate with poor database performance as the system relies heavily on real-time reporting that requires instance query responses. Manufacturing products' quality and production targets can be affected as the result of delayed queries. Therefore, the need to maintain the acceptable level of database performance in this domain is crucial. One task in maintaining database performance is identification and diagnosis of the root causes that may cause delayed queries. Poor query design has been identified as one major cause of delayed queries that affect real-time reporting. Nevertheless, as various methods available to deal with poor query design, it is important for a database administrator to decide the method or combination of methods that work best. In this paper, we present a case study on the methods used by a real manufacturing industry company called as Silterra and the methods proposed in the literature that deal with poor query design. For each method, we elicit its strength and weaknesses and analyse the practical implementation of it.

Key words: Query rewriting · Database performance tuning

INTRODUCTION

In modernization of technological era, the amount of data has been exploding in many application domains such as healthcare, public sector, retail and manufacturing. For example, in healthcare, electronic medical records consist of patient's family history, illnesses and treatment, which able to improve preventive care and disease treatment. In some cases, doctors take images with machines like a computerized axial tomography (CAT) or magnetic resonance imaging (MRI). These images can help map out patient's problems and help save lives. The needs to retain long-term active data or even permanent are increasing [1]. Data are accumulated and transformed to big dataset before they can be stored in a database. Big datasets can cause overhead to Database Management System (DBMS) and lead to database performance issues [2]. Database performance is a crucial issue, which can decrease the ability of the DBMS to respond to queries quickly.

Poor database performance cause negative consequences such as in financial, productivity and quality of the businesses in many application domains. In this research, we will focus on database performance issue in manufacturing domain. This research will base on a real case study in a semiconductor fabrication factory, Silterra Malaysia Sdn Bhd.

Silterra, like many other semiconductor manufacturing companies, is known to be the one of the most complex manufacturing operations. Most of the processes are delicate, for example the process of dicing the wafers must be carefully monitored, as the wafers are thin and fragile. Even a tiny scratch may scrap the wafer. These delicate processes require close monitoring, which is beyond human's capability. In addition, processing of 40,000 to 50,000 work-in-progresses (WIP) usually takes 50 to 70 days, 300 to 400 equipments and 300 to 900 steps to complete [3,4]. Manufacturing Execution System (MES) is used to manage WIP, equipment automation, material control system (MCS) routing and material transfer within

Corresponding Author: Nor'azah Md Khushairi, Software Engineering Department,
Faculty of Information and Communication Technology,
Universiti Teknikal Malaysia Melaka (UTeM), Hang Tuah Jaya, 76100,
Melaka, Malaysia.

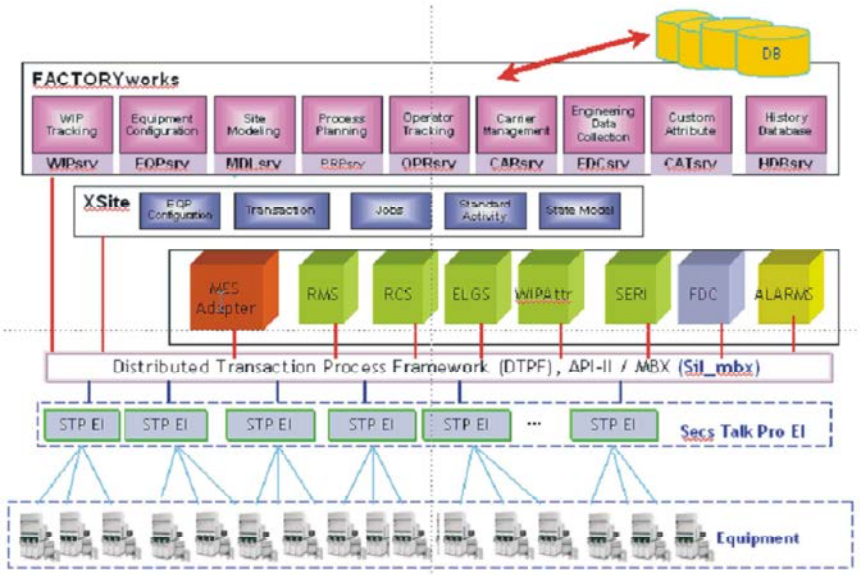

Fig. 1: Silterra MES Architecture [26]

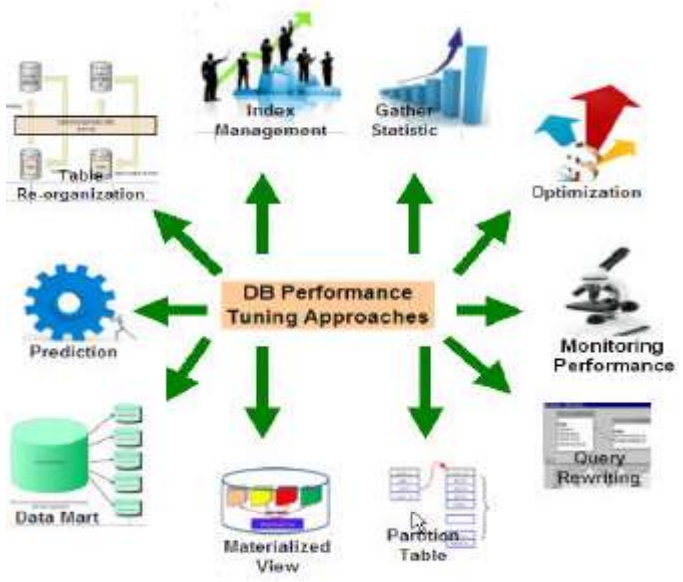

Fig. 2: Methods for Database Query Tuning

an automated material handling systems (AHMS). Huge amount of data are recorded automatically or semi automatically in multiple databases during fabrication process where the data become the input to monitor lot movements in the semiconductor fabrication plant, or shortly FAB. These data will be retrieved in timely manner to produce meaningful reports.

MES database composed of a collection of subsystems, each with a specific task that will interact with the user application programs and the database as shown in Fig.1. Complexities of the processes in FAB industry contribute to huge database because every single transaction needs to be recorded. Wafer transactions are characterized by a diverse product mix, re-entrant process flow, different processes types and different disruption [13]. As a result, processing overhead that must be dealt by Database Management System (DBMS) lead to database performance issues [2].

Silterra MES Architecture [26]

Literature Review: There are several factors that affect database performance such as database settings, indexes, memory, CPU, disk speed, database design and application design [8]. In general, database optimization

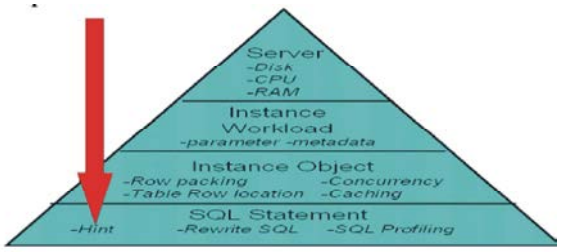

Fig. 3: DBMS tuning hierarchical level [11]

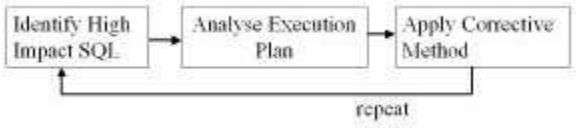

Fig. 4: Basic steps in SQL tuning

involves calculating the best possible utilization of resources needed to achieve a desired result such as minimizing process time without impacting the performance of any other system resource. It can be accomplished in three levels that is hardware, database or application.

Figure 3 shows DBMS database performance tuning hierarchical level. The tuning process should be performed from top to bottom as suggested by Burleson [11]. Database tuning should start with system level tuning because changes in other level might undo the tuned execution plan. However, application level can impose stronger influence on the database as compared to other levels in the hierarchy [6]. Due to this, it is important to monitor the SQL task and continuously estimate the remaining SQL execution time.

Basically, SQL tuning involves three steps as shown in Fig. 4. Firstly is to identify the problematic SQL that imposes high impact on the performance. Then, analyse the execution plan to execute a specified SQL statement with the cost. This is followed by rewriting the SQL by applying the corrective method. This process will be repeated until the SQL have been optimized [21].

Several methods have been used to improve the database performance as shown in Fig. 2.Generally DBA will perform most of these methods while application developer will perform the one that related to SQL only such as query rewriting. DBAs need to perform a regular inspection and database performance diagnosis. This process identifies the top ten activities that cause the performance problem. DBAs need to know exactly what is the cause of performance problems before they can apply corrective method.

Table 1 shows the methods that commonly used to deal with poor performance database caused by SQL queries, where for each method, weelicit the advantages and limitations.

Table 1: List of Methods for SQL Query Tuning

Methods	Literature	Advantages	Limitation
Gather statistic	For Oracle DB. It relies on up to date statistic to generate the best execution plan [19].	Updated statistics helps optimizer to select perfect execution plan for a query. For Oracle DB. It relies on up to date statistic to generate the best execution plan	Can be resource consuming. Must plan accordingly before executing
Index Management	Indexes are optional structures associated with tables and clusters that allow SQL statements to execute more quickly against a table [21].	Index created columns helps queries to select using index instead of doing full table scan, which is usually expensive.	DML statements can be slow if there is a lot of indexes on the table
Table Re organization	to improve the performance of queries or data manipulation language (DML) operations performed against these tables[22].	All data blocks will be moved to be together and prevent fragmentation which can cause slowness	Usually time consuming and needs downtime
Prediction	Estimating the Space Use of a Table, estimating the Space Use of an Index and Obtaining Object Growth Trends [21].	Able to predict the problem before it happened	Sometimes the prediction are not accurate because the data consumed is not increase in sequence
Data Mart	A data warehouse that is designed for data to be collected directly from various sources [21],	The database will be grouped according to schemas and departments. Easy to do maintenance and will improve the database performance.	Only applicable to schemas which are less than 100GB. It's not optimum to use data mart for bigger database.
Materialized view	A materialized view provides access to table data by storing the results of a query in a separate schema object [21].	Fast synchronization between source and target. Data can be refreshed on preferred method.	Complex queries on MV tables perform badly especially if there are joins with other tables
Partition Table	Splits the table into smaller parts that can be accessed, stored and maintained independent of one another [20].	1. Improve performance when selecting data 2. Easily for data pruning.	Hard to do maintenance for DBA's on partitioned table if it involves lots of partitions in a table. Index creation will be very slow if using hash partition
Query Rewriting	Query rewriting consists of the compilation of an ontological query into an equivalent query against the underlying relational database [10,23].	Improve the way data are being selected. By adding hints in the SQL, sometimes enhance the performance of individual queries.	Can be a troublesome job to change hardcoded queries. Queries that are not tested thoroughly could cause slowness.
Monitoring Performance	To determine possible problems, locates the root causes and provides recommendations for correcting them [21].	Able to identify the root cause of the problem.	Only DBA able to do monitoring.
Optimization	Query Optimization is the process of choosing the most efficient way to execute a SQL statement [10,21].	Helps queries to run faster. Data retrieval can be improved. Parameter tuning and memory tuning enable the database to perform in optimum level.	Must be done with supervision and wrong parameter set can cause database to go down or perform badly. Memory leak is also possible.

Table 2: SQL Queries Rewrite Technique

Findings	Rewrite Type	Researcher	Ref
ConQuer permits users to postulate a set of key constraints together with their queries. The system rewrites the queries to retrieve all (and only) data that is consistent with respect to the constraints.	Constraint	Fuxman *et al*., 2005	[28]
Techniques that can do the following. (a) Automatically rewrite programs to replace multiple calls to a query (b) Rewrite a stored procedure/function to accept a batch of bindings, instead of a single binding.	Binding	Ravindra *et al*., 2008	[29]
Develops a semantic query rewriting mechanism on heterogeneous database enabled web information system with complex ontology mapping technology.	Mapping	P.Liang *et al*., 2008	[30]
Propose a general query-optimization framework that treats regular and restructured views in a uniform manner and is applicable to SQL select-project-join queries and views with or without aggregation.	View comparisons	Chirkova *et al.,* 2006	[31]
Propose a consistency check that can handle a surprisingly large subset of SQL (Skolemization).- (=, =, <, >, ≤, ≥)	conjunctions	Brass *et al*., 2005	[32]
Tool support that is able to handle embedded queries in a wide range of host languages	embedded queries	Brink *et al.,* 2007	[33]
Discusses the techniques in SQL writing, tuning, utilization of index, data distribution techniques in a parallel processing DBMS architecture.	index	R.Nayem, 2013	[34]
Propose to augment SQL with set predicate two approaches– an aggregate function-based approach and -- a bitmap index-based approach. for optimizing queries with multiple predicates	predicate	Li Chengkai *et al.,* 2012	[35]
Novel concept of weak bindings that is essential for the queries containing outer joins.proposed a novel concept of join-reducibility	binding	Bhargava *et al*., 1995	[36]
Propose a practical approach for this optimization problem, called "coarse-grained optimization," complementing the conventional query optimization phase.	sequences	Tobias *et al*., 2003	[37]
Novel approach: zTuned is a system that formalizes and automates the process of SQL tuning using an experiment-driven approach.	smart exploration of the plan space.	Herodotos and Babu, 2009	[7]
Propose a novel canonical abstraction for outerjoin queries.	outerjoin	Rao *et al*., 2004	[38]
QShuffler is a query scheduler that takes query interactions into consideration in order to minimize the completion time of all workload and to improve the overall performance of the database.	multi-dimensional linear regression model	Brittany and Herodotos,2007	[39]

Identifying and fixing poor SQL performances can help to solve the performance problem as stated by Darmawan *et al.* [14]. Therefore, there is a need to fix problematic SQL queries by understanding the problem and re-writing them. It is expected that, queries that undergone the 're-write' process will retrieve accurate results within an acceptable time rate given tight CPU or IO constraints. We summarised SQL rewrite techniques proposed in the literature as shown in Table 2.

MES Reporting Problems: MES Reports plays an important role for the management to manage the FAB operation, to achieve the daily output target and to reduce cycle time. For example, if the reporting system is not available even for one hour, the line managers will have a difficulty to understand the output they can meet. In Silterra, one hour without reports can cause 2000 wafer moves lost which is equivalent to a few thousand USD lost. In addition, during wafer processing real-time reporting is important to know how many process wafer has been completed givena equipment that need to run multiple processes. Without real time reporting,a user can make an error in wafer processing.

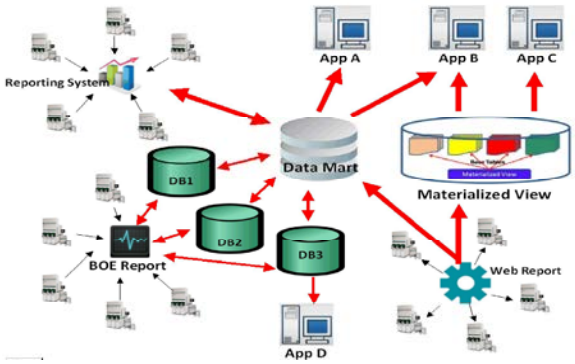

Fig. 5: MES Report architecture

Figure 5 below shows the MES report architecture which able to retrieve data from multiple database and produce multiple type of report. Data from various sources must be gathered and combined before a complete report can be produced, by issuing a set of queries. Data mart and materialized view have been created for faster data retrieval. The information will be refreshed hourly everyday. This information is crucial for the management personals to run the operation and to improve the cycle time in the FAB.

As MES has a complex architecture with multiple modules and databases with large sizes, one challenge is to respond to queries made by MES users. These queries are issued to produce real time monitoring reports for the managements and customers. MES reports that consists of information about WIP status and history movements starting from the product creation until the end of it. Data from various sources must be gathered and combined before a complete report produced by issuing set of queries. This information is crucial for the management personnel to run the operation and to improve the cycle time in the FAB. On the other hand, database performance monitoring helps (Database Administrator) DBAs and application developers to locate and identify the causes of bottlenecks due to poor database performance. SQL query is one major problem that affects MES reporting task and database performance as a whole.

Herodotou and Babu pointed out that an attempt to improve a poorly performing execution plan produced by the database query optimizer is a critical aspect of database performance tuning [7]. Identifying and fixing the poor SQL performances can help to solve the performance problem as stated by Darmawan *et al*. [14]. Therefore, there is a need to fix problematic SQL queries by understanding the problem and re-writing them. It is expected that, queries that undergone the 're-write' process will retrieve accurate results within an acceptable time rate given tight CPU or IO constraints within MES environment.

In the next section, we will present SQL tuning approaches adopted by Silterra.

SQL Tuning Approach

Identify Problematic SQL: Database administrator (DBA) normally will perform regular database inspection and performance diagnoses to identify any bottleneck in the DBMS. Problematic SQL can be identified by response time, CPU workload, I/O and wait class using Oracle tool such as SQL Trace and TKPROF. However, application developers have a limited access on the Oracle tuning tool but still need to do tuning. Therefore, SQL scripts as shown in Fig. 5 and 6 have developed and used to collect the SQL execution statistics in order to pinpoint problematic SQL.

The scripts need to be run at the application peak time to identify which SQL use high usage resource. When the performance issue occur, DBA and application developer need to response immediately and need to check the system to make sure the problem can be solve and no interruption to the FAB. The list of currently

```
SELECT s.machine, p.spid, SUBSTR(s.Program,1,25)
PROGRAM, s.Logon_Time, s.status ,
s.SQL_ADDRESS,s.terminal, s.TYPE, q.sql_text
FROM v$session s, v$process p,  v$sqlarea q
WHERE s.TYPE <>'BACKGROUND' AND s.paddr = p.addr
AND s.sql_address = q.address(+) ORDER BY s.Logon_Time
```

Fig. 5: To retrieve currently running query

```
SELECT s.machine, p.spid, s.Program, s.Logon_Time, s.status ,
s.terminal, s.TYPE,
q.EXECUTIONS,q.DISK_READS,q.BUFFER_GETS,
ROUND((q.BUFFER_GETS-q.DISK_READS)/q.BUFFER_GETS,2)
*100 hit,
ROUND(q.DISK_READS/q.EXECUTIONS,2) reads_per_run,
SQL_TEXT
FROM v$session s, v$process p,  v$sqlarea q
WHERE  s.TYPE <>'BACKGROUND' AND s.paddr = p.addr AND
s.sql_address = q.address(+) AND
q.EXECUTIONS>0 AND q.BUFFER_GETS>0  AND
(q.BUFFER_GETS-q.DISK_READS)/q.BUFFER_GETS<0.8
ORDER BY hit DESC;
```

Fig. 6: To retrieve low efficient SQL

running SQL can be retrieve by using the SQL in Fig 5. Based on logon time, we are able to know when this SQL start to run and able to identify whether it is a problematic SQL or not. Meanwhile Fig. 6 can be use to retrieve low efficiency SQL based on the hit ratio analysis. These scripts are using buffer get (logical I/O) and disk read (physical I/O). High buffer gets can cause high disk reads, which causes I/O workload increase and I/O path bottleneck [24]. In fact high buffer gets will increase the CPU time of the SQL.

SQL Tuning Methods: All the methods listed in the literature are the general methods use in Oracle environment and it is used in Silterra to improve the SQL performance bottleneck. The selection of which methods to be used is base on the type of the problems.

Gather Statistic: This is the most important methods for cost-based optimization approach to generate a statistics to calculate the selectivity of predicates and to estimate the cost of each execution plan. These activities are performing in a regular basis by DBA to provide the optimizer with the latest information about schema objects.

Optimization: This method will determine the most efficientexecution plan for the queries. The decision is made based on the statistical information from method 1. The optimizer will generate sub-optimal plans for some SQLstatements runs in the environment. The optimizer will choose the sub optimal plan based on the optimizer inputs such as object statistic, table and index structure, cardinality estimates, configuration and IO and CPU

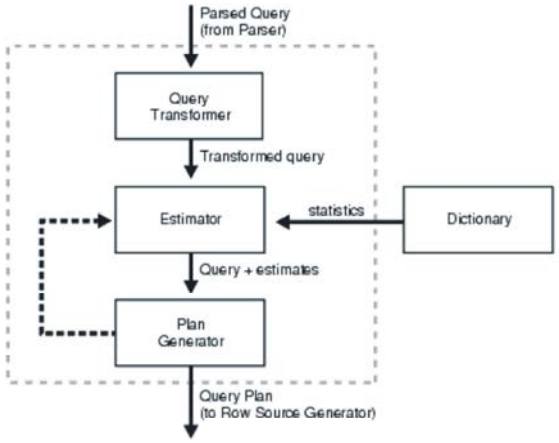

Fig. 7: Oracle Query Optimizer Component[21]

estimates. Figure 7 shows the query optimizer component and it's operation.

Index Management: Based on the statistic, DBA able to identify whether the sql is using full scan data selection or using index. Some of the case it is approriate to apply the index to the full scan table. Index exist primary to improve performance however the challenge is to select the wisely and right index. Selecting the right indices is a very difficult optimizationproblem: multiple candidate to choose, may benefit some parts of workload and incur maintenance overhead [27]. Indexes able to improve the performance significantly however we need be aware on the index cost. It is important to create appropriate index which able to improve the performance.

Monitoring Performance and Prediction: Frequent monitoring the system able to predict the problem before it happened. This method able to determine the possible problems and to locate the root causes of the problem. Furthermore, it will provide recommendations for correcting them.

Table Reorganization: This methods able to improve the performance of the query where all data blocks will be moved to be together and prevent fragmentation which can cause slowness. However we unable to perform this on the big production table since it needs system downtime and longer time needed based on the size of the table.

Data Mart: MES database size are huge, which will takes really long time to generate the history data and unable to produce immediate feedback for online reporting report. In this case, the implementation of data marts enable users to gain faster access to the common reports. In MES reporting, there are a need to keep hourly and daily snapshot data to produce the trend reporting charts. In this case data need to load into the data mart hourly and keep in the summarize manner for easier data retrieval. There are two report applications using the data mart concept which able to provide on line reporting to user. At the same time it able to reduce the traffic and bottleneck on the production database since user are not using the application that directly SQL to it. Data from various sources can be gathered and combined in the data mart to produce the summary report. However, it needs to properly design and maintain to avoid the problems in future.

Partition Table: MES tables are rapidly growing and huge data size is one of the common contributions to SQL bottleneck. In this case, the huge table will be analyst for partition where huge tables and index are split into smaller parts. In this example we split the table using transaction date column to partition the data by year. Each partition are independent object with its own name. Report performance will drastically increase because queries only will access fraction of the data based on the retrieval date selection. In fact, it only go to the specific partition table and less data to be scan and process. Therefore, it will be easier for data maintenance and faster data retrieval in order to overall manageability of database system [21]. At the same time maintenance including data purging, rebuilding index and backup can run faster, However, this method causes maintenance overhead for DBA in the case involving large amount of partitioned tables. In addition index creation will be very slow if hash partition is used.

Materialized View

Change the Commonly Use SQL to use Materialized View: MES Reports provide multiple types of hourly and daily reports in different type of format and layout for different purpose. In most cases, it is using the same raw data from multiple MES table. In this case materialized view (MV) is the best solution to get the snapshot data that is refreshed and synchronized from the source data and target at the specific time. All the related report will select from view and it will reduce the data selection from MES database, which can reduce the DBMS bottleneck. A MV is more complex than an index since MV may be define over multiple table which makes the problem of selecting materialized views significantly more complex

than that of index selection [25]. However, this method able to reduce repetitive I/O and unnecessary repeating large table full scan. During MV refresh, it will not affect the exactness and efficiency of SQL in applications [25]. MV needs to be monitored frequently to avoid database resource intensive due to MV auto refresh.

Rewrite SQL: SQL is a declarative language, which can be written in many forms to get the correct query results, however it produce different execution plan and response times. SQL query rewriting consists of the compilation of an ontological query into an equivalent query against the underlying relational database [23]. This process will improve the way data being selected and able to improve the performance drastically. However it can be a troublesome job to change hardcoded queries. Furthermore, queries that are not tested thoroughly could cause delay.By improving database performance through SQL queries rewriting, we can minimize the 'cost' that must be paid by those industries due to poor SQL queries. Below are the normally SQL rewrite method have been done:

Modify the SQL to Add Hints: Database optimization involves calculating the best possible utilization of resources needed to achieve a desired result such as minimizing process time without impacting the performance of any other system resource. Oracle optimizer will determine the most efficient way to execute a SQL based on the object and condition in the SQL [21]. However in some cases the SQL does not execute as expected. In this case, we need to rewrite the SQL and add hint in the SQL and it improve the SQL performance tremendously. In this environment the usage of optimizer hint able to improve the SQL performance drastically sometimes more than 100% improvement. In fact, application developers have more knowledge about the data that the optimizer. By providing the hints, it able to instruct the optimizer to choose a certain query execution plan based on the condition.

Shared SQL Statement: Shared SQL statement means using the same character class, same case and number of different space [9]. The example below show that these 3 queries cannot be share because Example 1 and 2 are not using the same case letter and example 3 contain space.

Example 1: SELECT * FROM FWTBL1
Example 2: SELECT * from FWTBL1
Example 3: SELECT * FROM FWTBL1

This methods able to reduce the DBMS workload by sharing the cache.

Union Issues: In MES reporting system, complex SQL have been created and most of important report are using UNION to combine the data from different source and purpose. UNION can slow down the performance According to shi the structure of the original SQL is optimized to remove the UNION operation, the greater effect and the COST reduced [9].The SQL need to be revised to enhance the effective way on using union or minimize the usage.

Use DECODE or CASE Statement Instead of Subquery: A subquery is a query within another query and a subquery also might contain another subquery. Subqueries can take longer to be execute than a join because of how the database optimizer processes them. In some cases, we need to retrieve the data from the same query set with different condition. Nested subqueries will do multiple queries based on the condition. These type of queries can be tune by rewrite it to use DECODE or CASE which will not do a multiple queries.

Driving Table: It is important to identify the driving table when there is a need to join more than one table in the FROM clause.Oracle parser will read from right to left means it will read the last table that we have specify. It is advise that we choose the smaller table as the driving table.

Automatic Tuning Tool: Recently, a lot of researches have been done and tools have been developed for autonomic tuning. Autonomic tuning tool are helpful but they cannot eliminate the need of physical tuning and does not scale well with query complexity [18].

Conclusion and Future Work: In conclusion, we presented the methods used by a real manufacturing industry company called as Silterra to deal with database performance problem caused by SQL queries and the methods proposed in the literature that deal with poor query design. The strengths and weaknesses of the methods have been presented where query rewriting is among the proposed methods used to deal with database performance issue. Even though several methods have been proposed for query rewriting (refer Table 2), none of the work describe how SQL query can be rewritten in real-time. Therefore, for our future work, will develop an algorithm of real-time SQL query re-writing for database

performance tuning. Technical issues those are barriers to implement successful real-time query rewriting will be explored. The result of this research will contribute to the database community as well as big data users where database performance is an issue. The usefulness of the technique proposed will be validated in a real manufacturing industry.

ACKNOWLEDGMENT

The authors would like to thank Universiti Teknikal Malaysia Melaka (UTeM) for their support especially to Faculty of Information and Communication Technology for their support and facilities along this research study.

REFERENCES

1. Zhou, H., Z. Yao and H. Xiao, 2011. "The Design Research on the Storage Structure for Large Amount of Data of the Database, Proceeding of the International Conference on Advance Mechatronic System," Zhengzhao. China, pp: 384-388.
2. Adam Jacobs, 2009. The pathologies of big data, Communications of the ACM, v.52 n.8, August 2009.
3. Rahim, S.A., I. Ahmad and M.A. Chik, 2012. "Technique to Improve Visibility for Cycle Time Improvement in Semiconductor Manufacturing, Proceeding of IEEE International Conference on Semiconductor Electronics (ICSE), pp: 627-630.
4. Puvaneswaran Balakrishna, M.A. Chik, Ibrahim Ahmad and Bashir Mohamad, 2011. "Throughput Improvement in Semiconductor Fabrication for 0.13µm Technology," Kota Kinabalu.Malaysia.
5. Balakrishna, P., M.A. Chik, I. Ahmad and B. Mohamad, 2011. "Throughput Improvement in Semiconductor Fabrication for 0.13µm Technology," in RSM2011 Proceeding.,Kota Kinabalu, Malaysia.
6. Beeler, D. and I. Rodriguez, 2002. Optimizing your Database Performance, BMC Software.
7. Herodotou, H. and S. Babu, 2009. Automated SQL Tuning through Trial and (Sometimes) Error. In Proc. of DBTest '09. ACM.
8. Connolly, T.M. and C.E. Begg, SQL: Data.
9. Manipulation in A.D., 2002. McGettrick (3rd Ed). Database Systems: A Practical Approach to Design, Implementation and Management, (pp: 110-155). Harlow: Addison Wesley.
10. Shi, Jiyuan, 2010. "Research and Practice of SQL Optimization in ORACLE." Information Processing (ISIP), 2010 Third International Symposium on. IEEE.
11. Trezzo and C. Joseph, 1999. Oracle PL/SQL Tips and Technique, Osborne:MsGraw-Hill.
12. Burleson, D.K., 2010. "Advanced Oracle SQL Tuning-The Definitive Reference, Oracle in Focus" 2nd ed, Rampant.
13. Rao, M. and S. Shah, 2002. Experimentation A Research Methodology.
14. Ponsignon, T. and L. Mönch, 2010. "Architecture for Simulation-Based Performance Assessment of Planning Approaches in Semiconductor Manufacturing," Neubiberg. Germany.
15. Darmawan, B., G. Groenewald, A. Irving, S. Henrique and M. Snedeker, 2003. Database Performance Tuning on AIX, IBM International Technical Support Organization, pp: 291.
16. Li, Dandan, Lu Han and Yi Ding, 2010. "SQL Query Optimization Methods of Relational Database System." Computer Engineering and Applications (ICCEA), 2010 Second International Conference on. Vol. 1. IEEE.
17. Cao, Bin, 2006. "Optimization of complex nested queries in relational databases." Data Engineering Workshops, 2006. Proceedings. 22nd International Conference on. IEEE.
18. Bruno, Nicolas and Surajit Chaudhuri, 2006. "To tune or not to tune?: a lightweight physical design alerter." Proceedings of the 32nd international conference on Very large data bases. VLDB Endowment.
19. Hu, Ling, *et al.,* 2008. "QueryScope: visualizing queries for repeatable database tuning." Proceedings of the VLDB Endowment, 1.2: 1488-1491.
20. Cao, Wei and Dennis Shasha, 2013. "Tuning in action." Proceedings of the 16th International Conference on Extending Database Technology. ACM.
21. Herodotou, Herodotos, Nedyalko Borisov and Shivnath Babu, 2011. "Query optimization techniques for partitioned tables." Proceedings of the 2011 ACM SIGMOD International Conference on Management of data. ACM.
22. Oracle, 2013.
Oracle Database Administrator's Guide 11g Release 1 (11.1).
23. Hobbs, L. and P. Tsien, 2005. Oracle Database 10g Release 2 Online Data Reorganization and Redefinition.
24. Gottlob, G., G. Orsi and A. Pieris, 2011. Ontological queries: Rewriting and optimization. Data Engineering ICDE, IEEE 27th International Conference, Hannover.

25. Optimize Database Performance. DB Speed (WWW page]. URL http://www.dbspeed.com/case_ study/ sql_with_high_buffer_gets.html.
26. Chaudhuri, Surajit and Vivek Narasayya, 2007. "Self-tuning database systems: a decade of progress." Proceedings of the 33rd international conference on Very large data bases. VLDB Endowment.
27. CIM Architecture, Silterra Technical Document, unpublish.
28. Schnaitter, Karl and Neoklis Polyzotis, 2012. "Semi-automatic index tuning: Keeping dbas in the loop." Proceedings of the VLDB Endowment, 5.5: 478-489.
29. Fuxman, Ariel, Elham Fazli and Renée J. Miller, 2005. "Conquer: Efficient management of inconsistent databases." Proceedings of the 2005 ACM SIGMOD international conference on Management of data. ACM.
30. Guravannavar, Ravindra and S. Sudarshan, 2008. "Rewriting procedures for batched bindings." Proceedings of the VLDB Endowment, 1.1: 1107-1123.
31. Liang, Ping, *et al.,* 2008. "Semantic Query for Integrated Heterogeneous Database Systems." Intelligent Networks and Intelligent Systems, 2008. ICINIS'08. First International Conference on. IEEE.
32. Chirkova, Rada and Fereidoon Sadri, 2006. "Query optimization using restructured views." Proceedings of the 15th ACM international conference on Information and knowledge management. ACM.
33. Brass, Stefan and Christian Goldberg, 2005. "Proving the safety of SQL queries." Quality Software, 2005. (QSIC 2005). Fifth International Conference on. IEEE.
34. Van Den Brink, Huib, Rob Van Der Leek and Joost Visser, 2007. "Quality assessment for embedded sql." Source Code Analysis and Manipulation, SCAM 2007. Seventh IEEE International Working Conference on. IEEE, 2007.
35. Rahman, Nayem, 2013. "SQL optimization in a parallel processing database system." Electrical and Computer Engineering (CCECE), 2013 26th Annual IEEE Canadian Conference on. IEEE.
36. Li, Chengkai, *et al.,* 2012. "Set Predicates in SQL: Enabling Set-Level Comparisons for Dynamically Formed Groups."): 1-1.
37. Bhargava, Gautam, Piyush Goel and Balakrishna R. Iyer, 1995. "Simplification of outer joins." Proceedings of the 1995 conference of the Centre for Advanced Studies on Collaborative research. IBM Press.
38. Kraft, Tobias, *et al.,* 2003. "Coarse-grained optimization: techniques for rewriting SQL statement sequences." Proceedings of the 29th international conference on Very large data bases-Volume 29. VLDB Endowment.
39. Rao, Jun, Hamid Pirahesh and Calisto Zuzarte, 2004. "Canonical abstraction for outerjoin optimization." Proceedings of the 2004 ACM SIGMOD international conference on Management of data. ACM.
40. Fasy, Brittany Terese and Herodotos Herodotou, 2007. "Dynamic Concurrency Control while Scheduling Query Mixes."