QUERY REWRITING USING MULTITIER MATERIALIZED VIEWS FOR CYBER MANUFACTURING REPORTING

N. M. Khushairi[1], N. A. Emran[2] and M. M. Yusof[3]

[1]Computer Integrated Manufacturing (CIM),
Silterra Malaysia Sdn Bhd, 09000 Kulim, Kedah, Malaysia.

[2]Computational Intelligence Technologies (CIT), Center of Advanced Computing Technologies (CACT), Universiti Teknikal Malaysia (UTeM), 76100 Durian Tunggal, Melaka, Malaysia.

[3]Sustainable IT-economics, Information Systems, Technology Management & Technopreneurship (SuITE), Center of Technopreneurship Development (C-TeD), Universiti Teknikal Malaysia (UTeM), 76100 Durian Tunggal, Melaka, Malaysia.

Corresponding Author's Email: [2]nurulakmar@utem.edu.my

Article History: Received 3 August 2017; Revised 5 October 2017; Accepted 8 December 2017

ABSTRACT: Within cyber manufacturing context, Internet of Data (IoD) technology has enabled manufacturing sector to store and transfer mass data rapidly for processing. Data growth which is driven by advancement in the way data are produced and interconnected has caused volume of data a crucial issue to address. As such, in monitoring delicate wafer processing in semiconductor manufacturing, reporting delay problem caused by databases of high data volumes is intolerable. This is because, various reports (that require access to large databases) need to be frequently generated in the shortest time possible. Reporting delay is usually handled through SQL query rewriting. In this paper, the results of experimenting SQL query rewriting by utilizing multitier materialized views structure is presented. In particular, we define sub-materialized views (SMVs) concept, and implement it using real data sets from SilTerra (a semiconductor industry). The outcome of the experiment supports the hypothesis that SQL query rewriting using SMV outperforms the classic rewriting. The results reveal that the performance of SMV is far better (than without SMV) for complex queries against large data sets. The benefits of SMV are not limited to cyber manufacturing domain as the use of SMV can contribute other industries with similar problem.

KEYWORDS: *Multitier Materialized Views; SQL Query Rewriting; Cyber Manufacturing; Cyber-Physical Systems; Industry 4.0*

1.0 INTRODUCTION

Manufacturing sector has been predicted to have fast growth, due to the growth of the “Internet of things" (IoT) that rely heavily on a great number of sensors, measuring, checking, and automatic regulating instruments [1]. In fact, in the era of Industry 4.0, this prediction has become a reality as manufacturing becomes a part of cyber-physical systems (CPS) today. Within CPS, cyber manufacturing is employed to utilize transformative technologies that translates data from interconnected system into operations that are predictable for resilient performance [2].

Empowered by Internet of Data (IoD), manufacturing sector is capable to store and transfer mass data rapidly for processing [3]. Nevertheless, the advancement in the way data is produced and transferred makes data volume a challenge. For example, a wafer semiconductor manufacturing industry like SilTerra Malaysia Sdn Bhd, to fabricate and to produce wafers within the shortest time possible is a goal [4]. As semiconductor processes are delicate and require close monitoring, sensors technologies are used to overcome human operators’ limitation [2]. These sensors are interconnected with the machines and hardware to collect data from a number of complex fabrication processes [5].

Manufacturing Execution System (MES) is used to monitor fabrication processes including work-in-process (WIP), equipment automation, process flow, and material control system (MCS) on a factory floor. About 50 to 70 days are needed to process 40,000 to 50,000 work-in-process (WIP); which involves 400 types of equipment and 300 to 900 steps to complete[4, 6-7]. The fabrication process complexity is characterized by a diverse product mix that is changing over time where different types of disruptions need to be handled [6]. These processes produce the raw data (collected largely by sensors) and also contribute to huge transaction history. These transaction history records are stored in several databases that keep growing over time. These records are extracted to produce reports for monitoring and predictive analytics. In addition, the fact that manufacturing has the largest amount of data has been highlighted in McKinsey [8] and in other studies [9-11].

With massive volumes of data, manual reporting is no longer an attractive option in cyber manufacturing environment. Not only the manual method is labour intensive, it is also inefficient in achieving the time-sensitive production goal [12]. Big data technology such as [13] can be used to aid massive data processing. Nevertheless, the ability to rapidly extract actionable data (in form of individual reports) is a pre-requisite for the next-stage data processing handled by Hadoop (or similar technologies) for predictive analytics preparation. Data that are extracted from databases through Structured Query Language (SQL) queries are used to generate reports. Using the large databases as source to generate reports, the problem that hinders rapid data extraction (and thus rapid reporting) in this industry is slow SQL queries.

To deal with reporting delay, a common action taken by Database Administrators (DBAs) is to tune these problematic SQL queries by rewriting them. In this paper, we present the results of implementing SQL rewriting using multitier materialized view concept. In particular, we propose to test the performance of SQL queries using sub-materialized views (SMVs) in speeding up reporting.

The rest of this paper is organized as follows. Section 2 of this paper presents the related works on the methods for database tuning where the focus is on SQL rewriting techniques. Section 3 provides the definition of SMV and describes the experiment conducted to evaluate SMVs. Section 4 presents and discusses the results of the experiment and Section 5 concludes the paper.

2.0 RELATED WORKS

Several factors that affect database performance include database settings, indexes, memory, CPU, disk speed, database design, and application design [14]. Database performance tuning is usually performed at: hardware level, instant workload and instant object or SQL statements. SQL is a declarative language, which is used in Database Management System (DBMS) to allow users to access and manipulate the database. SQL tuning which is an application level tuning imposes stronger influence on database performance as compared to other methods. SQL tuning is a difficult task as it requires identification of poor SQL instantly [15]. It is a challenging and time-consuming task, which needs experts to manage it [16]. These experts must possess thorough understanding of database technology, DBMS, application design, SQL query and related knowledge in this field. There are several methods proposed for SQL tuning, and the selection of the right method depends on the types of the problem.

A common routine performed by DBAs is to monitor and to identify problematic SQL queries in the system. These SQL queries will be rewritten as a way to tune them and this process usually involve laborious trial-and-error steps [17]. Adding hints in the SQL or index is one way to tune the SQL queries [18]. It is anticipated that queries that have undergone the rewrite process will retrieve accurate results within the given constraints (time, CPU or IO).

Ji et al. [19] proposed SQL query rewriting using the translation of ontological queries into equivalent queries against the underlying relational database. This process improves selection of data that consequently can improve the overall database performance. However, this approach can be a troublesome job as changes to queries need to be hard coded. Furthermore, queries that are not tested thoroughly could cause delay. In addition, SQL performance depends on the execution plan produced by database query optimizer [20].

Another common method used to deal with poor reporting performance is summarizing relevant data for reporting into a data mart. Data marts are able to manage and improve some of the report performance [21]. Instead of the direct query to the production database, the report application will retrieve the data from the data mart. There are scripts used to summarize the data on specific time within certain duration (such as every five minutes, hourly, daily or weekly). The data collected in the data mart are more organized and this can aid frequent access. Nevertheless, this method requires deep knowledge on the data structure to perform data aggregation on various dimensions. A set of SQL scripts must be developed up-front to cope with various table sizes and data structures.

Another method used to rewrite SQL queries is by using materialized view (MV). MV is a database object that keeps the query, runs the query at the specific time, and maintains the results of the query in the storage [19]. Even though the name contains view, it is not a virtual table like a view. It behaves like a table and sometimes referred to as a snapshot. Oracle defines MV as a replica of a target master from a single point in time where another master table continuously updates replication tables. MV data will be updated by refresh process upon changes to the base tables. This refresh process is performed incrementally or through complete refresh.

Several studies (as shown in Table 1) have reported the benefits of MVs in improving SQL queries speed, where in these studies, query response time is used as performance indicator. Studies involving MVs also covers the selection problems [22-25]. Nevertheless, none have attempted studying the benefits (or costs) of SQL query rewriting using the subsets of MVs. In this paper, we refer subsets of MVs as sub-materialized view (SMVs) that are created based on other MVs. To the best of our knowledge, the closest recorded idea of SMVs can be found in Oracle's online documentation, where the concept of multitier MVs is introduced [26].

Table 1: Studies in MVs' performance

Researcher Findings	References
A modified strategy of group query based on MV (GQMV) is proposed by making full use of the star schema feature to improve searching capability.	[16]
MV reporting functions are introduced which rewrite queries with reporting functions as well as aggregate queries.	[27]
Rewriting XPath Queries using MVs is proposed with an algorithm for finding minimal rewritings.	[28]
Answering queries using MVs with minimum size can reduce the size of the relations needed to compute the query answer. A method that efficiently finds a view set with a small size is proposed.	[28]

Multitier MVs concept allows MVs creation based on other MVs (subsets of MVs). Within the tiers, SMVs are defined and labelled according to their levels in the tiers, as shown in Figure 1. For instance, MVs that are based on level 1 MV are called level 2 MVs. Thus, within the tiers, SMVs exist from level 2 onwards. The benefit of using multitier MVs has been recorded for replication, to support organizations that have limited network and storage resources [26]. The question of whether the use of SMVs in query rewriting can speed up the query performance or otherwise.

3.0 METHODOLOGY

In this paper, we propose SQL Query rewriting using SMVs with the aim to deal with reporting delay in cyber manufacturing context. Essentially, MVs are created using base tables (BTs) as their source, and SMVs are created using MVs as their source. In [26], the term master MV is used for MVs that are used to create their SMVs. In fact, MV at any level can be a master MV.

These database objects (BTs, MVs and SMVs) are related and their relationship can be visualized in a Venn diagram as shown in Figure 2. For simplicity, the diagram depicts the case of a master base table and a master MV. In a more complex case, multiple master base tables (and MVs) are supported, for example, SMV created based on two or more master MVs.

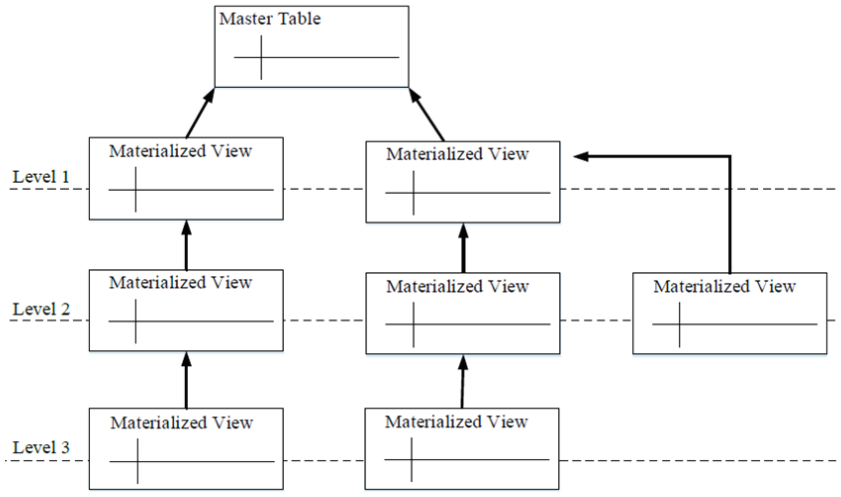

Figure 1: Levels of multitier materialized view [26]

Suppose that SMV is a set of tuples of a SMV; *MV* is a set of tuples of a MV; *BT* is a set of tuples of a BT. The relationship among these three database objects is defined as: *SMV* is a proper subset of MV and *MV* is a proper subset of BT, which is denoted as $SMV \subset MV \subset BT$. SQL queries created using BT, MV and SMV are denoted as: query against BT (QBT), query against MV (QMV) and query against SMV (QSMV).

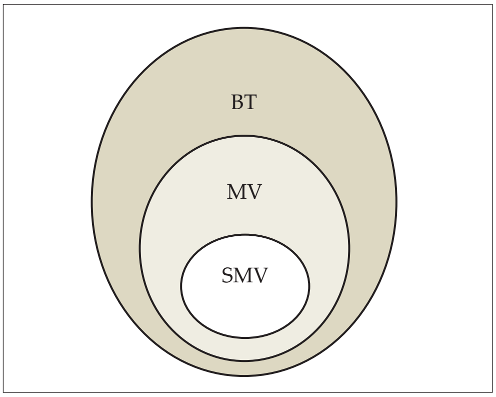

Figure 2: A Venn diagram for relationships among database objects

To evaluate SMVs, experimental approach used by similar works [27, 30] is used to test the hypothesis that QSMV performs better than QMV. Figure 3 shows the flow of the experiment. SQL query speed (elapsed time) is chosen as the variable to be observed in the experiment. The problematic SQL queries, in form of QBT, are identified from the delayed reports that appear on MES monitoring dashboard.

As problematic QBTs can be found involving many tables, in this experiment, we choose Factory Works Historical WIP Database where it consists of huge tables. The largest table is WIP transaction table (about 430GB), that consists of 38 columns, with 1000,000,000 tuples. WIP is the main table used to keep wafer's transaction history, where most problematic QBTs are found using this table. Other tables used are WIP Step History table (20GB) and Comment table (84GB). For variety, we also used smaller tables that are Lot table (179MB) and CAT table (6.09MB). These tables become the source (BTs) of QBTs under measure.

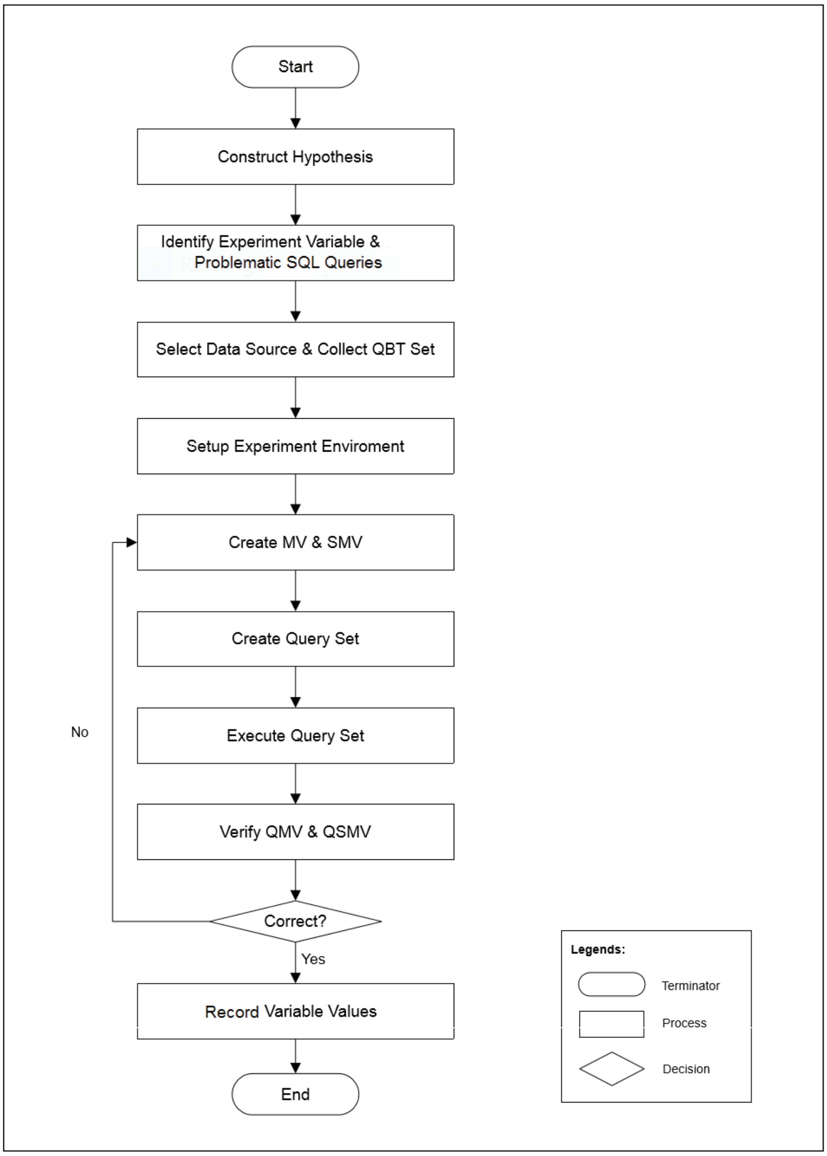

Figure 3: The flow experiment for SMV evaluation

Once the QBT set is ready, we setup the experiment environment where a Fujitsu M5000 workstation with 3 CPU of SPARC64 VII 2.6 GHz and 32 GB memory on the MES database server, set on Oracle 11g platform. Then, QMV set and QSMV set are created, after the MVs and SMVs for these queries are created. 14 query sets are used in this experiment, where each set of query consists of: a QBT, a QMV and a QSMV. Thus, the total number of query used in this experiment is 42 queries. As it is important to ensure the validity of all query sets, in this experiment, the result (set of tuples) of each query set is used as the validity measure. Suppose that for query set i and Qi, are

- the result of QBT i, QBT_i is QBT_i ',
- the result of QMV i, QMV_i is QMV_i ',
- the result of QSMV i, $QSMV_i$ is $QSMV_i$ '.

Q_i is defined as valid if it satisfies the following equations:

$$|QBT_i'|=|QMV_i'|=|QSMV_i'| \tag{1}$$

$$\frac{|QBT_i' \cap QMV_i' \cap QSMV_i'|}{|QBT_i' \cup QMV_i' \cup QSMV_i'|}=1 \tag{2}$$

An example of a query set written in SQL is as shown in Table 2. SQL statements used to create MV and SMV are also shown in the table. Following similar step in [17], each set of query is executed three times and the average value of the elapsed time for the query items is taken. Once all 14 query sets are executed and verified, the results (SQL query elapsed time) are collected and prepared for analysis. Oracle utilities TRACE file and TKPROF are used to record the results.

Table 2: SQL Statements for Query Sets Definition

Query Sources	**Query Set**

BT :
WIPTRANSACTION
	QBT:
	SELECT txndate,productname, activity, COUNT(activity),SUM(lotqtyout)
	FROM WIPTRANSACTION
	WHERE txntime >= to_char(sysdate-1,'YYYYMMDD HH24MISS') and
	txntime <= to_char(sysdate,'YYYYMMDD HH24MISS')
	GROUP BY txndate,productname, activity;

MV: TXNWIP_MV	**MV definition:** CREATE MATERIALIZED VIEW TXNWIP_MV AS SELECT txndate,activity,productname,lotqtyout, txntime FROM wiptransaction WHERE txntime >= to_char(sysdate-1,'YYYYMMDD HH24MISS') AND txntime <= TO_CHAR(SYSDATE,'YYYYMMDD HH24MISS') GROUP BY txndate,productname, activity; **QMV:** SELECT txndate, productid, activity, count(activity), sum(lotqtyout) FROM TXNWIP_MV group by txndate, productid, activity;

SMV: TXNWIP_SMV	**SMV definition:**
	CREATE MATERIALIZED VIEW TXNWIP_SMV
	(txndate,productid,activity,countlotid,sumqtyout)
	AS SELECT txndate,productid, activity,COUNT (lotid),SUM (lotqtyout)
	FROM TXNWIP_MV GROUP BY txndate, productid, activity;
	
	QSMV:
	SELECT txndate,productid,activity,countlotid,sumqtyout FROM
	TXNWIP_HMV;

4.0 RESULTS AND DISCUSSION

Figure 4 shows the line graph that plots the elapsed time (in seconds) for the query sets under measure. The *x* axis represents the 14 query sets under measure (labelled as Q1-Q14); the first *y* axis (left) represents the elapsed time variable (in seconds) for QBT and the second *y* axis (right) represent the elapsed time variable (in seconds) for QMV and QSMV. Dual *y* axes are used to aid separating the range of elapsed time that is obviously significant for QBTs.

As expected, QBT set takes longer time to execute (as compared to QMV and QSMV set) because QBT set is of problematic SQL queries. Thus, in this experiment, we are more interested with the performance of QSMV relative to QMV's.

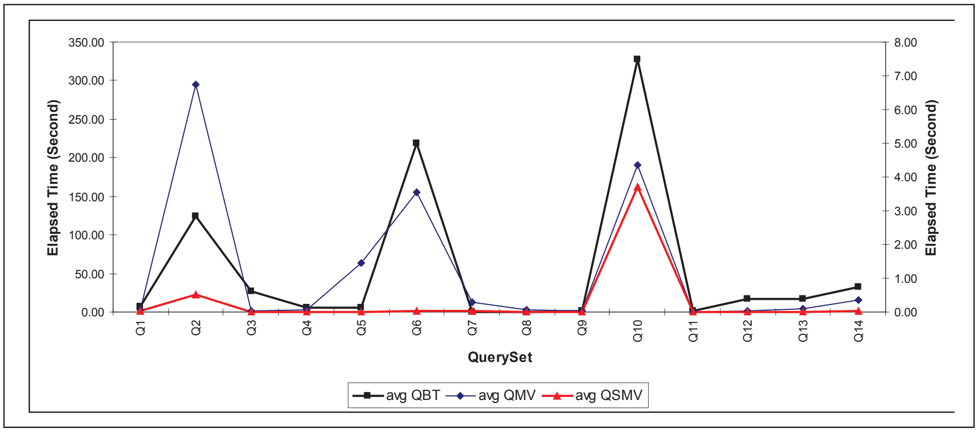

Figure 4: Elapsed Time Scores for Query Sets

The trend line shows that QSMVs takes less time to be computed as compared to QMVs in all sets. Particularly, QSMV performed significantly faster for Q2 and Q6 as compared to QMV (with the difference of 6.24 and 3.53 seconds respectively). This situation indicates the complexity of the QMVs in the set (as JOIN and aggregate functions are used), which requires longer time for SQL processing. With the use of SMV, the processing of the complex queries can be avoided as the queries have been computed in advanced (through SMV creation).

On average, the time taken for QSMV to complete is less than 0.3 seconds, with the maximum period of 3.71 seconds for Q10 (while QMV and QBT take about 4.35 seconds and 327.18 seconds respectively for Q10).

Marginal difference in elapsed time between QMV and QSMV can be found, however, in those cases, QSMV still leads the score. This situation can be described by the difference between the size of MV and SMV is small. In addition, less time is taken for Q1, Q8 and Q11 as the size MVs and SMVs for these query sets are small (less than 1 GB).

While the results yielded for the experiment reported in this paper favour QSMV, other aspects must be taken into account in evaluating the benefits of SMV in SQL query rewriting. Question regarding what are the ‘costs’ of using SMVs must be answered, as by knowing the effort needed to setup and maintain the SMVs we can decide on the acceptability of the trade-offs (if any).

5.0 CONCLUSION

In conclusion, this paper presented a proposal of SQL query rewriting by using SMVs to cope with reporting delay problem within cyber manufacturing context. The concept of SMV which has been defined in this paper is motivated by multitier MVs idea that is of limited coverage on SMV's contribution in SQL rewriting. The hypothesis of the experiment that SQL query rewriting using SMV performs better than SQL query rewriting using MVs has been supported by the result of the experiment. The outcome of the experiment also suggests that the performance of QSMV is far better than QMV especially in the case where complex queries are used and the size of MVs are significantly bigger than the size of SMVs. In the future, SMVs should be evaluated further especially in terms of the cost of implementing them. The maintenance aspect of SMVs also is an open problem especially in the case involving multiple master MVs.

ACKNOWLEDGMENTS

We acknowledge financial assistance received from the Ministry of Education Malaysia under Fundamental Research Grant (FRGS/1/2015/ICT04/FTMK/02/F00289) and the support from Universiti Teknikal Malaysia Melaka.

REFERENCES

[1] Economic Planning Unit. (2014). *Complexity analysis study of Malaysia's manufacturing industries* [Online]. Available: http://www.epu.gov.my/en/content/final-report-research-complexity-analysis-study-malaysias-manufacturing-industries.

[2] J. Lee, B. Bagheri, and C. Jin, "Introduction to cyber manufacturing," *Manufacturing Letters* , vol. 8, pp. 11–15, 2016.

[3] L. Wang and G. Wang, "Big Data in cyber-physical Systems, digital manufacturing and Industry 4.0," *International Journal of Engineering and Manufacturing (IJEM),* vol. 6, no. 4, pp. 1–8, 2016.

[4] T. Ponsignon and L. Monch, "Architecture for simulation-based performance assessment of planning approaches in semiconductor manufacturing," in the Proceedings of Winter Simulation Conference (WSC), Baltimore, 2010, pp. 3341–3349.

[5] S. Munirathinam and B. Ramadoss, "Big data predictive analtyics for proactive semiconductor equipment maintenance," in IEEE International Conference on Big Data (Big Data), Washington, DC, 2014, pp. 893–902.

[6] S. R. A. Rahim, I. Ahmad, and M. A. Chik, "Technique to improve visibility for cycle time improvement in semiconductor manufacturing," in 10th IEEE International Conference on Semiconductor Electronics (ICSE), Kuala Lumpur, 2012, pp. 627–630.

[7] P. Balakrishna, M. A. Chik, I. Ahmad, and B. Mohamad, "Throughput improvement in semiconductor fabrication for 0.13μm technology," in IEEE Regional Symposium on Micro and Nano Electronics, Kota Kinabalu, 2011, pp. 228–231.

[8] Mckinsey Global Institute. (2011). *Big data: The next frontier for innovation, competition, and productivity* [Online]. Available: https://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/big-data-the-next-frontier-for-innovation.

[9] D. Boyd and K. Crawford, "Critical questions for big data: Provocations for a cultural, technological, and scholarly phenomenon, "*Information, Communication and Society,* vol. 15, no. 5, pp. 662–679, 1986.

[10] R. Dubey, A. Gunasekaran, and S. Childe, "The impact of big data on world-class sustainable manufacturing," *The International Journal of Advanced Manufacturing Technology*, vol. 84, no. 1-4, pp. 631–645, 2016.

[11] J. Lee, H. Kao, and S. Yang, "Service innovation and smart analytics for industry 4.0 and big data environment," *Procedia Cirp*, vol. 16, pp. 3-8, 2014.

[12] T. Wilschut, I. J. B. F. Adan, and J. Stokkermans, "Big data in daily manufacturing operations," in the Proceedings of the Winter Simulation Conference, Savannah, GA, 2014, pp. 2364–2375.

[13] J. Lee, E. Lapira, B. Bagheri, and H. Kao, "Recent advances and trends in predictive manufacturing systems in big data environment," *Manufacturing Letters*, vol. 1, no. 1, pp. 38–41, 2013.

[14] L. Hu, K. A. Ross, Y.-C. Chang, C. A. Lang, and D. Zhang, "QueryScope: visualizing queries for repeatable database tuning," in the Proceedings of the VLDB Endowment, vol. 1, 2008, pp. 1488–1491.

[15] D. DeHaan, D. Toman, M. P. Consens, and M. T. Ozsu, "A comprehensive XQuery to SQL translation using dynamic interval encoding," in the Proceedings of the ACM SIGMOD international conference on Management of data, California, 2003, pp. 623–634.

[16] L. Guodong, W. Shuai, L. Chang'an, and M. Quanzhong, "A modifying strategy of group query based on materialized view," in 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE), Chengdu, 2010, pp. V5-381-V5-384.

[17] H. Herodotou and S. Babu, "Automated SQL tuning through trial and (sometimes) error," in the Proceedings of the Second International Workshop on Testing Database Systems, Providence, RI, 2009, pp. 1–6.

[18] D. Taniar, H. Y. Khaw, H. C. Tjioe, and E. Pardede, "The use of Hints in SQL-Nested query optimization," *Information sciences* , vol. 177, no. 12, pp. 2493–2521, 2007.

[19] S. Ji, W. Wang, C. Ye, J. Wei, and Z. Liu, "Constructing a data accessing layer for In-memory data grid," in the Proceedings of the Fourth Asia-Pacific Symposium on Internetware, QingDao, 2012, pp. 1–7.

[20] J. Cohen, B. Dolan, M. Dunlap, J. M. Hellerstein, and C. Welton, "MAD Skills: New analysis practices for big Data," in the Proceedings of the VLDB Endowment, vol. 2, 2009, pp. 1481–1492.

[21] L. Lim and B. Bhattacharjee, "Optimizing hierarchical access in OLAP environment," in IEEE 24th International Conference on Data Engineering, Cancun, 2008, pp. 1531–1533.

[22] R. Goswami, D. K. Bhattacharyya, and M. Dutta, "Materialized view selection using evolutionary algorithm for speeding up big data query processing," *Journal of Intelligent Information Systems.* vol. 49, no. 3, pp. 1-27, 2017.

[23] P. P. Karde and V. M. Thakare, "Selection of materialized view using query optimization in database management : An efficient methodology," *International Journal of Management Systems IJDMS*, vol. 2, no. 4, pp. 116–130, 2010.

[24] X. Li, X. Qian, J. Jiang, and Z. Wang, "Shuffled frog leaping algorithm for materialized views selection," in 2nd International Workshop on Education Technology and Computer Science, New Jersey, 2010, pp. 7–10.

[25] P. Bagale and S. R. Joshi, "Optimal materialized view management in distributed environment using random walk approach," *Journal of Advanced College of Engineering and Management* , vol. 1, pp. 67–73, 2016.

[26] R. Urbano. (2008). *Materialized view concepts and architecture-Oracle Docs* [Online]. Available: https://docs.oracle.com/cd/ B28359_01/ server.111/ b28326.pdf

[27] D. Habich, W. Lehner, and M. Just, “Materialized views in the presence of reporting functions,” in 18th International Conference on Scientific and Statistical Database Management (SSDBM’06), Vienna, 2006, pp. 159–168.

[28] Y. Xu and S. Hu, “QMapper: A tool for SQL optimization on hive using query rewriting,” in the Proceedings of the 22nd International Conference on World Wide Web, Rio de Janeiro, 2013, pp. 211–212.

[29] R. Chirkova, C. Li, and J. Li, “Answering queries using materialized views with minimum size,” *The VLDB Journal-The International Journal on Very Large Data Bases*, vol. 15, no. 3, pp. 191–210, 2006.

[30] L. M. Wein, “Scheduling semiconductor wafer fabrication,” *IEEE Transactions Semiconductor Manufacturing*, vol. 1, no. 3, pp. 115–130, 1988.