Optimizing Database Query Performance Using Table Partitioning Techniques

Khaled Saleh Maabreh
Faculty of Information Technology
Zarqa University
Zarqa, Jordan
kmaabreh@zu.edu.jo

***Abstract*—Database performance is a primary branch of information technology which deals with the proper management of the database. The primary goal of the data management is to provide the organizations daily routines with powerful applications that have to be run efficiently. Performance optimization has a critical role in improving the database usage. In particular, managing the massive amount of data generated daily by various users.**

The main target of this research is to evaluate the data partitioning technique in enhancing the performance of quires submitted to large databases. The encouraging results show that data partitioning could improve the performance of DBMS which manages massive databases. The experiments reveal that data partitioning has a remarkable impact on the query execution time in big databases, which exceeds 35% compared to small database size or not partitioned database.

Keywords—performance, big data, optimization, data partitioning, database management.

I. INTRODUCTION

Database performance is a primary branch of information technology that deals with the proper management of the database, the main aim of this field is to provide the organizations daily routine business with powerful applications that have to be run as efficient as possible. This aim can be achieved through various monitoring methods to track different weak points that can cause failures in database productivity. Different factors may affect the database performance such as database structure and design, implementation, physical database design, and log file distribution; such these factors may degrade the database performance if not wisely configured[1]. Today, due to the advanced technology of digital sensors and communications, people and system can generate large size of data which adds more overhead on the DBMSs. The last two decades have seen a growing trend towards large data sizes of various types that can be hosted in a database. Currently, Petabytes of different types of data are moving over the internet. By the year 2025, the data generated by the Internet is predicted to surpass the brain size of everyone living in the world[2]. Therefore, more powerful solutions and efficient algorithms are urgently needed to cope with processing and storing massive databases.

All information systems depend mainly on data management. Utilizing efficient techniques in data management would result in productive and more powerful database systems. Currently, the information revolution has a direct impact on information systems capacities. Therefore, the task of organizing and manipulating this massive amount of data becomes a big challenge face the database administrators and designers. Enhancing performance is considered as the most critical aspect in managing information systems. This aspect is the primary objective for the database designers and the fundamental requirement for the end users. Therefore, the performance aspects have to considered in all system development phases, in addition to improving the physical database design by query performance and indexes management[3, 4].

Partitioning techniques can provide a remarkable improvement on database performance for multi-table joins. This improvement comes from breaking up a large join into smaller joins where the DBMS can retrieve its content faster than the large join. Therefore, the breaking technique can be applied whenever a partition key is joining two or more tables. Furthermore, partitions provide objects independence which leads to the high availability of data. This independence comes from the availability of multiple data alternatives in various partitions.

Generally speaking, any table has primary data including LOB attributes and get partitioned and stored separately from the LOB, the update operations executed over the primary data could run faster. LOB is a Large Object data type that may contain images or documents within a database. For example, a student record may include a photo image which does not frequently changed. However, the student details (such as address, major, college, and so on) could change. Moreover, faster storage is used for the student record[5, 6].There are many techniques of data partitioning to control how data is distributed and placed into individual partitions[5, 6]. For example, Oracle offers three fundamental data distribution methods: List, range, and hash partitioning techniques. List partitioning technique depends on the partitioning key by specifying a list of discrete values. Range partitioning is based on ranges of values of the partitioning key that can be defined for each partition. Finally, hash partitioning depends on a hashing algorithm that maps data to partitions. Fig.1 shows a simple representation example of these techniques.

978-1-7281-0385-3/18/$31.00 ©2018 IEEE

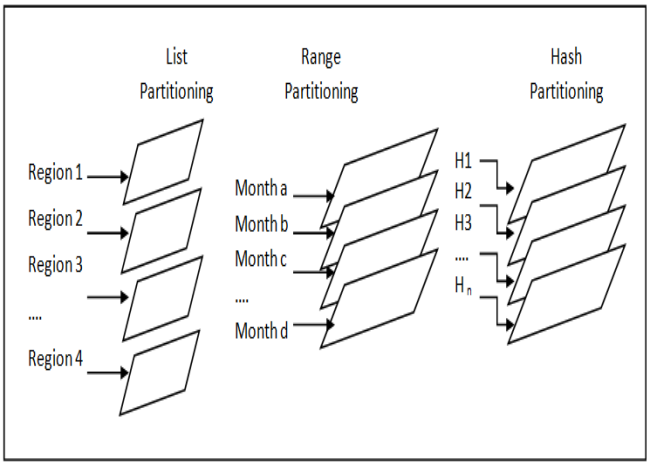

Fig. 1. Data partitioning techniques

II. Related Works

In recent years, thousands of publications have been published in the area of big data. Much of these researches have focused on identifying and evaluating the performance of queries conducted against the database. For obtaining an optimal performance for database applications, the table-partitioning technique has been used by [7]. A set of queries are executed on different database sizes using three partitioning strategies. The results showed that the query performance has significantly improved on the partitioned table compared to not partitioned ones. Another research by Herodotou, et al. [8] has shown that the partition optimization techniques reduces the overhead and overcome the plans produced by current optimizers. This research also introduced a newly proposed technique that generates an efficient query by using multi-way joins over partitioned tables. Results produced by their model overcome the previous methods of optimization. Agrawal, et al. [6] suggest a new method for integrating the large space required in physical design for data partitioning with managing constraints. Their suggested technique is being used for designing a scalable solution to the integrated environment. This integration could enhance the database performance and make the managing of a database as easy as possible.

On the other hand, and in a distributed database, there is a high need to enhance the query performance; this urgent need is increased day by day because of huge data distributed over several sites. This requirement can be achieved by accessing the minimum amount of data to utilize the network efficiently and to minimize the cost. For enhancing the query performance in a distributed database, the breaking up of a database row into logically interrelated attributes instead of the whole row is proposed by Maabreh and Al-Hamami [9]. More detailed studying of query performance is given by [4]. They present and analyze new techniques for primary and derived horizontal data partitioning. Data localization and propagation with the effect of database performance is also discussed and analyzed. Their analysis showed the benefits of how the horizontal data partitioning could improve the query processing.

In summary, it has been shown from this review that the data partitioning technique has a significant impact on the query performance. This impact can be observed when most of the executed queries are of the read-only mode and running against large data.

III. Methodology

A database consisting of three tables is created using Oracle 10g. The database schema is described as follows:

Table 1: StudentInfo (Stno,AdminYear, Fname,Mname, Lname,FacultyID, Major, St_Address, Gender, StStatus, DoB)

Table 2: CourseInfo (CourseID, Title, Hours, FacultyID, DeptID, CourseType)

Table 3: Schedule(CourseID, Section, Year, Term, FacultyID, Room, MaxCapacity, ActualNumber, SecDays, SecDaysGroup, SecTime, InstructorID)

The student information table is partitioned according to the admission year by using a hash technique. This method will produce four partitions (from the year 2014 to the year 2017) so that each partition consists of students who are admitted in the same year. The course information table is not partitioned because it contains a small data size, while the Schedule table is partitioned by hash according to the SecDaysGroup. Regarding the section days group, it is coded to "1" if the days are Sunday, Tuesday, and Thursday. Code "2" will be used for Monday and Wednesday and finally, code"3" is used for the master degree courses that are offered on Saturday. In case of lab courses that have one credit hour, it will be coded to "1" if the section date is on Sunday or Tuesday or Thursday. Otherwise, it will be coded to "2".

Initially, these tables are filled with the data extracted from the currently used database at Zarqa University, There are 7000 students in the University, and more than 400 different courses are offered from different departments. Data size within these tables is too small to evaluate and compare the performance. So that, further data collection was required to determine exactly how the data partitioning is affecting the query performance. Therefore, virtual data with more than 100,000 students and more than 5000 courses have been generated to fill these tables. Table 1 shows the data distribution among partitions for the student information table, the partitioned data for the Course-Sections table has shown in Table 2.

A set of queries is prepared to be executed against the database as shown in Table 3. Most of the queries are using the inner join, and some of them are using the outer join. They designed to retrieve database tuples according to the conditions that combine the partitioning key. The average response time and the average turnaround time are used as metrics to measure the system performance. The same set of queries will be executed on the database without partitioning as the first phase. In the second phase, the queries will be executing on partitioning tables with real data of small size. Finally, the execution will be on partitioning tables with virtual data of large size.

TABLE 1: NUMBER OF ROWS IN EACH PARTITION FOR STUDENT INFORMATION TABLE

Table Size	Total # of Rows (No Partitioning)	Partition Size by Year			
		2014	*2015*	*2016*	*2007*
Phase 1	7000	1730	1952	2104	1214
Phase 2	100000	18730	21305	24230	35735

TABLE 2: NUMBER OF ROWS IN EACH PARTITION FOR SCHEDULE TABLE

Table Size	Total # of Rows (No Partitioning)	Partition Size by Sec-Days Group		
		Group 1	*Group 2*	*Group 3*
Phase 1	400	235	131	34
Phase 2	5000	2875	1743	382

The purpose of executing the queries at three phases is to evaluate and to compare the system performance. Moreover, to show the data size and its effect on query performance.

IV. RESULTS AND DISCUSSION

The queries listed in Table 3 are executed on the database with varying workloads, and then the average response time and the average turnaround time are collected and analyzed. The process was repeated several times to increase the reliability and integrity of the received data.Fig.2 shows the average response time for the three phases. Phase one uses the data as it is with no partitioning. In the second phase, the data of small size is partitioned. Finally, the same tables are filled virtually with larger data size and then used in the queries execution. With more than 20% enhancement, the experiments on gathered data show that the data partitions have a big impact on the overall performance than non-partitions.

Furthermore, data partitioning for large volume data as a workload is outperformed the non-partitions with more than 35% improvement. Therefore, and whenever the data size is big, data partitions become a critical requirement for enhancing the performance. Moreover, Fig.3 shows a remarkable superiority of the turnaround time for the partitioning data over the non-partitioned, especially in large data size. The turnaround time superiority indicates and supports the easily secure of management and maintenance of data within a database. The partitioning can provide high benefits to a broad set of applications by enhancing performance, security, and data availability. Partitioning also provides the ability to the database designers and administrators to solve difficult problems for the systems that need high availability requirements.

TABLE 3. QUERY LIST

Query No.	Query Description
1	Select course number and course name for the courses that are registered by students whose numbers are greater than or equal to 20160001.
2	Select course number, section number and the time for all courses that held on Monday and Wednesday after 3:30 pm.
3	Select all students names whose registered in a course starting at 8:00.
4	Select the course number and name for all closing courses on Monday and Wednesday (Maximum Capacity = Actual number).
5	Select all not registered students.
6	Select all not enrolled courses.
7	Select student number, student name, and the course name for all graduate students (Master degree students).
8	Select student number, student name, and the course name for all students whose registered in lab courses.
9	Select all courses that are taken by students whose number between 20140001 and 20159999.
10	Select all courses that are not taken by students whose numbers are less than 20149999.

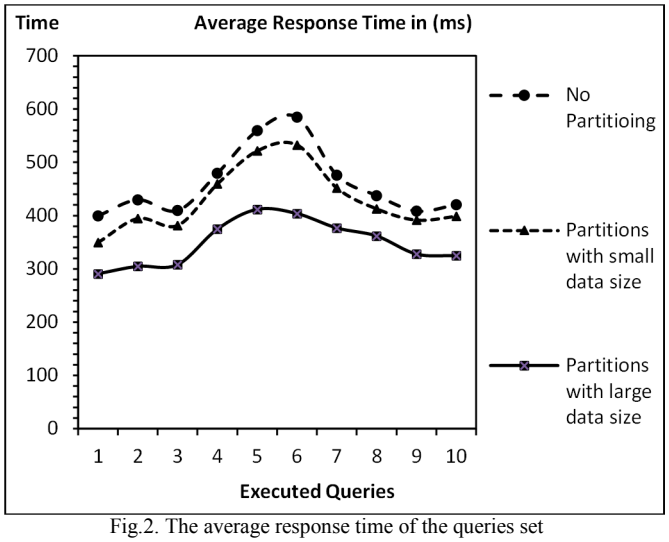

Fig.2. The average response time of the queries set

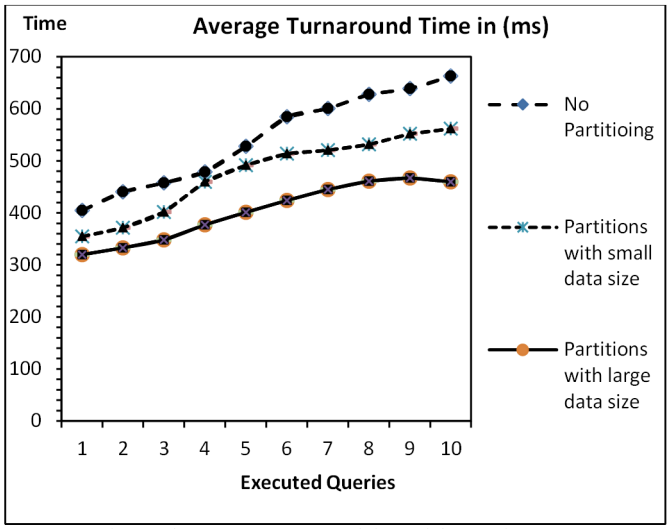

Fig. 3. The average turnaround time of the queries set

V. CONCLUSIONS

The purpose of this research was to determine the database management as an essential factor that can affect the performance of the executed queries. Such management control includes the keys, indexes and the data size. For evaluating the query performance, a big data has been divided into small fragments to be used in the executed queries. The findings provide observations into the effectiveness of the data partitioning on the query performance. The analysis of the data collected from running queries over a partitioned data tables confirms that the average response time is improving with more than 35% by comparing the query performance with none partitioning data, the result becomes significant in large data size. These findings complement those of earlier studies.

REFERENCES

[1] C. S. Mullins, *Database Administration: The Complete Guide to DBA Practices and Procedures*. USA: Addison-Wesley, 2002, p. 928.

[2] M. D. A. Praveena and B. Bharathi, "A survey paper on big data analytics," in *2017 International Conference on Information Communication and Embedded Systems (ICICES)*, 2017, pp. 1-9.

[3] C. CIOLOCA and M. GEORGESCU, "Increasing Database Performance using Indexes," *Database Systems Journal,* vol. 2, no. 2, pp. 13-22, 2011.

[4] L. Bellatreche, K. Karlapalem, and A. Simonet, "Algorithms and Support for Horizontal Class Partitioning in Object-Oriented Databases," *Distrib. Parallel Databases,* vol. 8, no. 2, pp. 155-179, 2000.

[5] H. c. Oracle. (2018, Aug-1-2018). *Database VLDB and Partitioning Guide*. Available: https://docs.oracle.com/database/121/VLDBG/

[6] S. Agrawal, V. Narasayya, and B. Yang, "Integrating vertical and horizontal partitioning into automated physical database design," presented at the Proceedings of the 2004 ACM SIGMOD international conference on Management of data, Paris, France, 2004.

[7] S. Matalqa and S. Mustafa, "The Effect of Horizontal Database Table Partitioning on Query Performance," *International Arab Journal of Information Technology (IAJIT),* Article vol. 13, no. 1A, pp. 184-189, 2016.

[8] H. Herodotou, N. Borisov, and S. Babu, "Query optimization techniques for partitioned tables," presented at the Proceedings of the 2011 ACM SIGMOD International Conference on Management of data, Athens, Greece, 2011.

[9] K. Maabreh and A. Al-Hamami, "Implementing New Approach for Enhancing Performance and Throughput in a Distributed Database," *International Arab Journal of Information Technology (IAJIT),* Article vol. 10, no. 3, pp. 290-296, 2013.