

A Novel Approach to Improve the Performance of the Database Storing Big Data with Time Information

Murat Tasyurek


Abstract— Big data is defined as data sets that are too large and/or complex to be processed by classical data processing methods. Big data analysis is essential because it enables more competent business movements, more efficient operations, and higher profits by using the data of institutions and organizations. However, large data sets are difficult to analyze because they are produced quickly, require large storage areas in computer systems, and the diversity of their data. In this study, a new approach using the denormalization method is proposed to accelerate the response time of the database in database systems where large volumes of data containing historical information are stored. Denormalization is defined as the process of adding rows or columns that are not needed to increase the reading performance of the database to the database system that has been normalized. In the proposed approach in this study, a large-volume data set consisting of real spatial data belonging to Kayseri Metropolitan Municipality (KBB), containing temporal information and having approximately 96,000,000 row records, was used. In the proposed approach, the response time of the query is accelerated by recording the time information as numbers to increase the query performance of large volumes of data recorded in date format due to the temporal query process. The performance of the proposed method is compared with the performance of the normalization method using actual data on Microsoft SQL Server and Oracle database systems. The method proposed in the experimental evaluations shows that it works approximately eight times faster. In addition, the experimental results showed that the proposed method improves query performance more than the normalization-based method as the data size increases.

Index Terms— Database performance, Denormalization, Large volume data, Temporal query.

I. INTRODUCTION

WITH THE widespread use of technology, data with different attributes are formed in many application areas [1]. For instance, data sent by smartphones and sensors, surveys, publicly shared comments (images, videos) on

MURAT TAŞYÜREK, is with Department of Computer Engineering of Kayseri University, Kayseri, Turkey, (e-mail: m_tasyurek@hotmail.com).

 <https://orcid.org/0000-0001-5623-8577>

Manuscript received Jan 17, 2022; accepted Aug 16, 2022.
DOI: [10.17694/bajece.1059070](https://doi.org/10.17694/bajece.1059070)

websites are data with different attributes [2]. If data is produced quickly, needs large storage areas, or consists of data with various features, it shows big data characteristics [3]. Big data is defined as data sets that are too large and/or complex to be processed by classical data processing methods [4]. Big data is usually stored in databases and analyzed using software specifically designed to handle large and complex data sets [5]. Big data analysis is vital because it enables more competent business movements, more efficient operations, and higher profits by using the data of institutions and organizations [6]. However, large data sets are difficult to analyze because they are produced quickly, require large storage areas in computer systems, and the diversity of their data [7].

The data production rate, size (volume), and diversity of big data do not mean anything on their own [8]. For the data set to be useful or usable, it must be transformed into information using various techniques. To analyze the data, it must be saved in a file or database system in a specific format. Database systems are widely used because they are designed to easily organize, store and retrieve large amounts of data [9]. The normalization method known as relational database theory is used in database design [10]. Normalization, also known as parsing, parses a table consisting of many columns and rows into subsets with fewer rows and columns to avoid duplication [11]. However, the increase in the number of records and tables in the database increases the response time of the database. To accelerate the response time of the database system to the queries, the process of accelerating the response time of the database by adding rows or columns that are not needed after the normalization process in the database design is defined as denormalization [12, 13]. In this study, a new denormalization-based method was proposed to accelerate the analysis of the data stored in the relational database, which consists of actual spatial data belonging to the KBB, contains time information, approximately 96,000,000 row records. In this study, the performance of the proposed novel approach is compared with the performance of the normalization-based method by storing the actual data in Oracle and MS SQL database systems.

A. Contributions

- In this study, a new approach is proposed to improve the performance of database querying in which large

volumes of data containing time information are stored.

- The performance of the proposed approach is compared on MS SQL and Oracle database systems using actual data from KBB.

B. Scope and Outline

- The main purpose of this study is to develop a new query technique for large databases containing space and time information with fast growth and large volume. For this reason, examining other features such as the variety that make up big data is out of the scope of this study.

In the following parts of this study, a literature review is presented, problems and approaches are introduced, the results of the experiments are discussed, and the results are shared.

II. RELATED WORKS

A database is an organized system of structured information and/or data stored digitally (electronically) in computer systems [14]. Database systems are widely used because they make it easy to access, manage, modify, update, control, and organize data [15]. To use these features of the database system and to work quickly, table designs must be created in a certain order. To ensure data integrity in the database system, dividing the data into sub-tables and to save them relationally is defined as normalization. However, as the number of tables and data increases in database systems, the response times of the queries used to extract information from the database get longer. To speed up the search process in database systems, indexing the searched columns using one of the indexing methods provided by the relevant database system is a widely used technique [16, 17]. However, since the indexing process requires a lot of storage space in computer systems, it cannot be used in cases where the storage space is limited. On the other hand, Online Analytical Processing (OLAP) method is an additional software technology that provides information by combining, grouping, or combining the attributes in databases and is used for rapid extraction of information from database systems [18, 19]. The denormalization method, known as accelerating the response time of the database by adding rows or columns that are not needed in the database system, is used instead of the OLAP system, which requires additional costs [20]. The denormalization method is a widely used technique to improve the performance of the database system [21-24]. On the other hand, NoSQL database systems, which have flexible schemas for modern applications and are specially designed for certain data models, have become widespread due to their practical use, ease of development, and performance [25, 26]. NoSQL databases use different data models, including graph, key-value, document, in-memory search [27]. On the other hand, NoSQL databases are designed for various data access patterns involving low latency applications [28, 29]. For this reason, NoSQL database

systems are not preferred in cases where low latency is important.

In addition, many devices that make up the transportation system are constantly generating data. For this reason, the data obtained from the devices used in the transportation system is considered big data. Special solutions need to be developed to overcome the problems running in the transport system. For example, Vela et al. [30] focused on the problem of designing the NoSQL document database with which to manage the information concerning accessible routes obtained by means of crowdsourcing techniques. Asaithambi et al. [31] proposed the microservice oriented big data architecture incorporating data processing techniques, like predictive modelling for achieving smart transportation and analytics microservices required towards smart cities. Gonzalez et al. [32] investigated into the testing of transactional services in NoSQL databases in order to test and analyses the data consistency by taking into account the characteristics of NoSQL databases for efficiency and velocity.

In this study, a new approach is proposed to overcome the problem of low latency in NoSQL database systems and to additional costs of OLAP systems, and to improve the date-based filtering and sorting performance of large volumes of spatial data containing time information.





III. APPROACHES

A. Basic Problem

The problem arose in keeping and analyzing the location information of the movements of the vehicles operating in the KBB public transportation system. In the KBB public transportation fare collection system, payments are made by considering the number of kilometers traveled by the vehicles along the route. However, traffic density, etc. Due to the reasons, the start and end times of the flights do not fully comply with the planned departure times. On the other hand, some of the delays during the voyage hours are caused by errors or omissions caused by the driver. Whatever the reason, it is vital to detect and solve the problem since the delays in the public transportation system affect the citizens using this system. To analyze a voyage in the public transport system, the locations sent by the vehicle before the start time of the journey, the locations sent during the trip, and the location information sent after the end time of the voyage can be determined together. To carry out this analysis, the location information (latitude and longitude), line information, speed, vehicle side number, and time information sent by the vehicles working in the public transportation system were recorded in the database system used by the KBB using the normalization method. In Fig. 1, the voyage start time, voyage duration, number of passengers carried, end time, and location information sent along the route of a vehicle with side number 387 operating on route 563 belonging to the KBB transportation system are presented on the map. The icons shown in Fig. 1 represent the following meanings:



Fig.1. Display of vehicle location information on the map

-  : The map equivalent of the GPS position sent by the transportation vehicle
-  : Bus station
-  : Line route
-  : Direction of line route

When the map layer and icons presented in Fig. 1 are examined in detail, it is seen that the bus with side number 387 is on the route of line 563. However, when it is desired to explore the accuracy and details of the voyage start and end times of the relevant vehicle, it is necessary to examine the locations it has sent before the route starts, instead of only the locations it sends along the route. In this case, the locations sent by the vehicle according to the vehicle side number and time information can be queried from the database system, and information about the vehicle can be obtained as a result of examining the relevant records on the map.

B. Normalization Based Approach

In database systems, the normalization process is defined as parsing a table with too many rows and columns into subsets of fewer rows and columns to eliminate repetitions [33, 34]. The structure of the database table called VEHICLE_LOCATION, which is created as a result of the Normalization process for the bus side number, line number, time information up to the second detail, point location information consisting of latitude and longitude values, the number of passengers in the vehicle and the speed information of the vehicle sent by the vehicles belonging to the KBB public transportation system. It is presented in Table I.

TABLE I
Information of VEHICLE_LOCATION Table

Column name	Column description	Data type
Vehicle ID	Side number to distinguish vehicles from each other	Integer
Route ID	Number of the route where the vehicle is running	Integer
Date of data	Sending time of GPS location information	Date time
Location	Point position of the vehicle (latitude and longitude)	Geometry (point)
Number of Passengers	Total number of passengers in the vehicle	Integer

VEHICLE_LOCATION table presented in Table I is a database table where the information sent instantly by all vehicles working in the public transportation system is stored. Queries run for analysis operations work on this table. For the voyage information presented in Fig. 1, the query sentences for MS SQL and Oracle database systems, which show all the information sent by the vehicle 10 minutes before and 10 minutes after the voyage, in chronological order and run in the table created with the normalization process, are shown in Fig. 2 (a) and 2 (b), respectively. In the query shown in Fig. 2 (a) and 2 (b), since the text expression to date field conversion functions of MS SQL and Oracle database systems and the parameters taken by the functions are different, the results of the queries are the same, but the query sentences differ from each other. In the MS SQL database system, the convert function is used to convert the text expression to date format, and the to date function is used in the Oracle database system [35]. In the MS SQL database system, the expression 120 in the convert function means “yyyy-MM-dd hh:mm:ss”.


```
SELECT * FROM VEHICLE_LOCATION
WHERE VehicleID=387
AND (DateOfData BETWEEN
CONVERT(DATETIME, '2020-12-31 13:14:09',
120)
AND
CONVERT(DATETIME, '2020-12-31 14:12:44',
120)
)
order by DateOfData
```

(a) MS SQL temporal query

```
SELECT * FROM VEHICLE_LOCATION
WHERE VehicleID=387
AND (DateOfData BETWEEN
TO_DATE(DATETIME, '31.12.2020 13:14:09',
'DD.MM.YYYY HH24:MI.SS')
AND
TO_DATE(DATETIME, '31.12.2020 14:12:44',
'DD.MM.YYYY HH24:MI.SS')
)
order by DateOfData
```

(b) Oracle temporal query

Fig.2. Temporal query

In the queries shown in Fig. 2 (a) and 2 (b), “yyyy” includes year information, “MM” shows month information, “dd” shows day information, “hh” - “HH24” show hour information, “mm” - “MI” show minute information, and “ss” shows seconds.

Although an index was created for the Vehicle ID and date fields of the VEHICLE_LOCATION table, which was created as a result of the normalization process in MS SQL and Oracle database systems, the running time of the queries presented in Fig. 2 (a) and 2 (b), which were run for the analysis of the public transport system, did not decrease at the desired level. Therefore, a novel approach has been proposed to speed up the running time of the query run with the normalization method and, therefore, to reduce the response time of the queries run for public transport system analysis operations.

C. Proposed Novel Approach

Accelerating the response time of the database by adding rows or columns that are not needed to a database system that has been normalized is defined as denormalization [22, 23]. As a result of the normalization processes, the database table described in detail in Table I was created, and the queries presented in Fig. 2 (a) and 2 (b) were run on this table. However, since the response time of the queries is very long, the analysis process takes longer. To overcome this problem, a new column of digit data type named Date_Number has been

added to the database table presented in Table I as part of the denormalization process. In the new column called Date_Number, it is suggested to keep the numerical equivalent of the date-time data format in the Date field. To convert date fields to numeric values, the functions are shown in Fig. 3 (a) and 3 (b) created for MS SQL and Oracle databases systems, respectively.

When only a new record is added to VEHICLE_LOCATION table, the date field is converted to numeric values using these functions and recorded in the Date_Number field. The numeric value of the date field in the format YYMMDDHH24MISS was created by only two digits of the year information YY, the month information MM, the day information DD, the hour information HH24, the minute information MI and the second information SS. The query of the denormalization-based method is presented in Fig. 4.

The query presented in Fig. 4 shows the query format shown in Fig. 2 (a) and 2 (b) converted to MS SQL and Oracle databases after denormalization. In the query shown in Fig. 4, the Date_Number field created after the denormalization process is used instead of the date field. Filtering and sorting are done according to the Date_Number field, a number field, instead of a date field. As in the normalization method, an index is created for the Date_Number field in the denormalization-based method.

```
CREATE OR ALTER FUNCTION [dbo].[FN_DATE_AS_NUMBER](@Time DATETIME)
Returns bigint
AS
BEGIN
Declare #number_of_Time varchar(20)
Declare #resultNumber bigint
BEGIN
SET #number_of_Time = convert(varchar, @Time, 120) + convert(
varchar, #number_of_Time, 8)
SET #number_of_Time=REPLACE(#number_of_Time, ':', '')
SET #resultNumber = convert(bigint, #number_of_Time)
END
return #resultNumber
END
```

(a) Function to convert MS SQL date format to numeric value

```
create or replace function FN_DATE_AS_NUMBER(P_DATE DATE ) RETURN NUMBER
IS
resultNumber NUMBER ;
number_of_Time varchar2(100) := '';
BEGIN
number_of_Time := to_char(P_DATE, 'YY')||to_char(P_DATE, 'MM')||
to_char(P_DATE, 'DD')||to_char(P_DATE, 'HH24')||to_char(P_DATE, 'MI')
||to_char(P_DATE, 'SS');
resultNumber :=to_number(number_of_Time);
RETURN resultNumber ;
END ;
```

(b) Function to convert Oracle date format to numeric value

Fig.3. Function to convert date format to numeric value

```

SELECT * FROM VEHICLE_LOCATION
WHERE VehicleID=387
AND (Date_Number BETWEEN
      FN_DATE_AS_NUMBER('201231131409')
      AND
      FN_DATE_AS_NUMBER('201231141244')
)
order by Date_Number

```

Fig. 4. Query of the proposed method

IV. EXPERIMENTAL EVALUATIONS

This study was carried out using the actual data of the vehicles working in the KBB public transportation system for December 2020. In Table II, the number of records of the entire data sets obtained from the KBB and the storage areas occupied by this data. In the KBB public transportation system, the communes sent by the vehicles for a month consist of 96,000,000 data sets. These data, which are sent instantly by hundreds of vehicles on a per-second basis, show big data characteristics because they are sent continuously and take up a lot of memory. Furthermore, since there is latitude and longitude information in the incoming data and it is known at which point the vehicle is on the geography, this data is also spatial data. Also, the data set is temporal since a vehicle sends data continuously at certain time intervals. Thus, the data set stores the historical data of the vehicles together with the time information.

TABLE II
DATA SETS

Sequence ID	Record Count	Size (megabyte)
1	100,000	551
2	250,000	1,380
3	500,000	2,765
4	1,000,000	5,546
5	2,000,000	11,136
6	3,000,000	16,815
7	4,000,000	22,622
8	5,000,000	28,595
9	10,000,000	57,647
10	15,000,000	87,508
11	20,000,000	118,428
12	25,000,000	150,640
13	30,000,000	184,383
14	40,000,000	250,638
15	50,000,000	320,316
16	60,000,000	393,989
17	70,000,000	472,260
18	80,000,000	555,784
19	90,000,000	645,265
20	96,000,000	712,803

Experimental evaluations sought answers to the following questions:

- What are the runtimes of normalization based and recommended approaches on incremental data set in MS SQL database?
- What are the runtimes of normalization based and recommended approaches on incremental data set in Oracle database?

To evaluate the performance of the methods in the experimental evaluations, the queries detailed in Fig. 2 (a), 2 (b) and 4 were run for 20 different vehicles operating in the KBB public transport system, and the average times were examined as the working time the methods. The experiments in this study were carried out on a desktop computer with Intel i7 11700 2.5GHz (8 Core), 32 GB Ram, 4 GB QUADRO graphics card, 2 TB SATA DISK, and Windows 10 Pro operating system installed.

A. Running Times of Methods in MS SQL Database System

In this experiment, the normalization and the result of the proposed method were examined in the MS SQL database system using the data sets detailed in Table II. In Table III, the working times of the methods are presented in seconds, and in Fig. 5, the operational times of the methods are examined in minutes. As the number of records in the data set increases, the working time of the methods increases.

TABLE III
RUN TIMES OF METHODS IN MS SQL DATABASE SYSTEM

Sequence ID	Normalization Based Method (millisecond)	Proposed Method (millisecond)
1	3	3
2	5	4
3	12	10
4	28	15
5	64	26
6	90	32
7	161	64
8	254	118
9	584	240
10	874	270
11	1,175	310
12	1,574	340
13	1,807	390
14	2,400	430
15	3,101	466
16	3,751	524
17	4,400	572
18	5,027	604
19	5,913	722
20	6,215	809

When Table III and Fig. 5 are examined in detail, there is almost no difference between normalization and denormalization methods up to 1,000,000 data sets. However, in cases where the data set has more than 10,000,000 rows of records, the response time of the query run with the normalization method exceeds a second. If the size of the data set is 96,000,000, the response time of the normalization-based method is over 6 seconds, which is a very long time for a system that works in real-time and is analyzed instantly. On the other hand, as the size of the data set increases, the working time of the normalization-based method increases much more than the proposed method. Although the proposed method's performance increases as the data set's size increases, a linear increase cannot be observed. The proposed method can run analysis queries in a very short time frame, such as 0.8 seconds, even in the worst case where the data set has

96,000,000 records. When the queries shown in Fig. 2 (a) and Fig. 4 are examined in detail, not only filtering but also sorting is performed. In the denormalization-based method, although it is not needed in the database design, the Date_Number column is added. The data in the Date column is converted to number format with the function presented in Fig. 3 (a). This column, which was added as part of the denormalization method, accelerated the performance of the database by approximately eight times for 96,000,000 data sets.

denormalization-based method. In the Oracle database system, although the performance of the proposed denormalization-based method increases as the size of the data set increases, a linear increase cannot be observed. The proposed method can run analysis queries in a very short time frame, such as 0.7 seconds, even in the worst case where the data set has 96,000,000 records. In the Oracle database system, the Date_Number column added within the scope of the denormalization method accelerated the performance of the

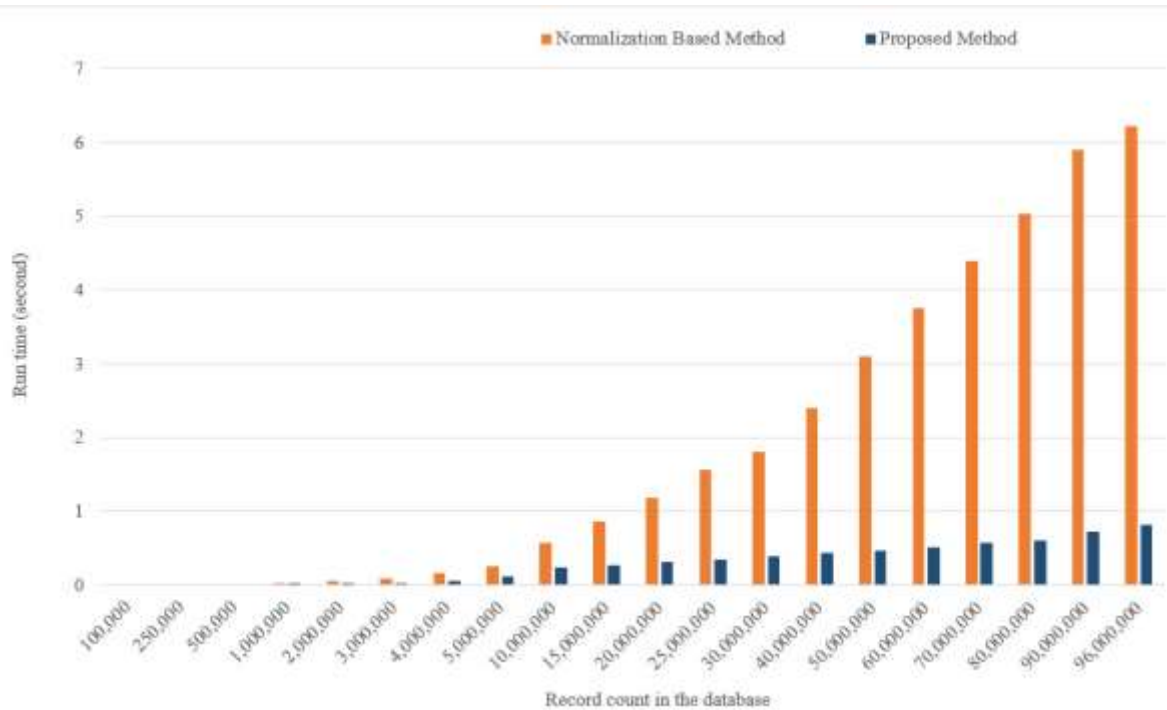


Fig. 5. Run times of methods in MS SQL database system

B. Running Times of Methods in Oracle Database System

In this experiment, the normalization and the result of the proposed denormalization-based method were examined in the Oracle database system using the data sets detailed in Table II. In Table IV, the working times of the methods are presented in seconds, and in Fig. 6, the working times of the methods are examined in minutes. As in the MS SQL database system, the operating times of the methods increase as the number of records in the data set increases in the Oracle database system. Similar to the MS SQL database system, when Table IV and Fig. 6 are examined in detail, there is almost no difference between normalization and denormalization methods up to 1,000,000 data sets. However, unlike the MS SQL database system, the response time of the query executed with the normalization method exceeds a second when the data set has over 20,000,000 rows of records. If the data set size is 96,000,000, the response time of the normalization-based method is over 5 seconds, which is a very long time for a system that works in real-time and is analyzed instantly. Similar to the experiments performed in the MS SQL database system, as the data set's size increases, the normalization-based method's running time increases much more than the

database to 96,000,000 data sets, approximately eight times as in the MS SQL database system.

TABLE IV
RUN TIMES OF METHODS IN ORACLE DATABASE SYSTEM

Sequence ID	Normalization Based Method (millisecond)	Proposed Method (millisecond)
1	2	2
2	4	3
3	9	8
4	24	14
5	55	21
6	80	31
7	150	39
8	270	52
9	500	60
10	810	95
11	1,030	120
12	1,300	150
13	1,600	190
14	2,150	250
15	2,700	310
16	3,300	380
17	3,800	430
18	4,300	490
19	4,980	588
20	5,311	658

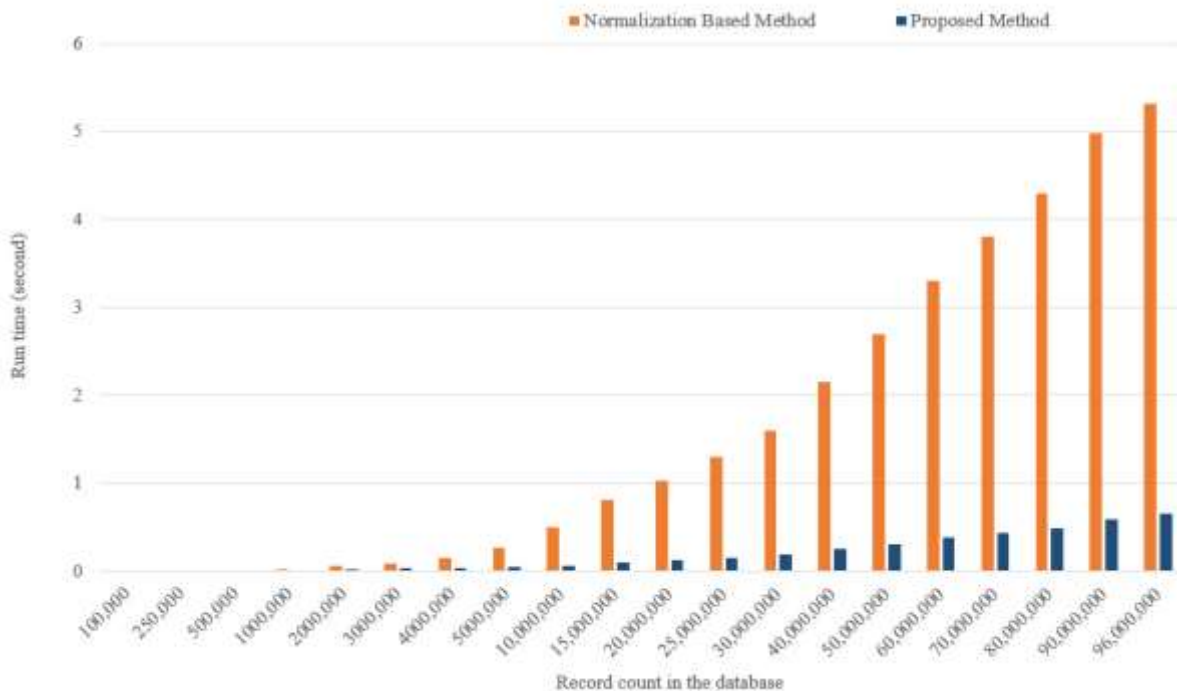


Fig. 6. Run times of methods in Oracle database system

C. Application

The application developed within the scope of this study is actively used by the KBB. It is used for route violation, stop violation, and control of the journey time of the lines operating in the public transportation system. Thanks to the proposed application, the list of lines that were violated is automatically presented, and by clicking the detail button, how the violation was made is displayed on the map thanks to the proposed application. Before the proposed practice was used, the detection of violations was only detected upon complaint. Thanks to the application, the detection of violations can be viewed instantly. The screenshot of the application used is presented in Fig. 7 and Fig. 8 in the Appendix part. Fig. 7 shows the list of voyages operating according to important criteria such as line name, number, start time, and end time. Fig. 8 presents the list of violating lines. In Fig. 8, the word GI indicates route violation, the word DI indicates to stop the violation, and the word HD indicates the line number changes while the voyage continues.

V. CONCLUSION

Big data systems can be used in transportation, banking, communication, media, entertainment, healthcare, education, manufacturing, etc. It is widely used in many different fields. Due to big data as the oil, the low latency of NoSQL database systems and the additional cost of data stored in relational database systems, and showing big data characteristics to OLAP systems are big problems. This study proposes a new

approach to improve date-based filtering and sorting performance in relational database systems where large volumes of spatial data containing time information are stored. The performance of the proposed method is examined using actual data in MS SQL and Oracle database systems. Experimental evaluations Since normalization and denormalization-based methods work very closely in data sets with less than 1,000,000 record numbers. There is no need for a denormalization-based approach in such data sets. However, as the amount of data in the data set increases, the working time of the normalization-based method increases more than the denormalization-based method. When the data set with 96,000,000 records is examined, the denormalization-based method responded eight times faster than the normalization-based method, since it performs both filtering and sorting operations numerically. Experimental evaluations have shown that as the number of records in the data set increases, the denormalization-based method works much faster than the normalization-based method.

ACKNOWLEDGMENT

We would like to thank the KBB for sharing large volumes of spatial data containing real-time information of the vehicles operating in the public transportation system used within the scope of this study. In addition, this study was supported by the Scientific Research Projects Coordination Unit of Kayseri University within the scope of project #FHD-2021-1043.

APPENDIX

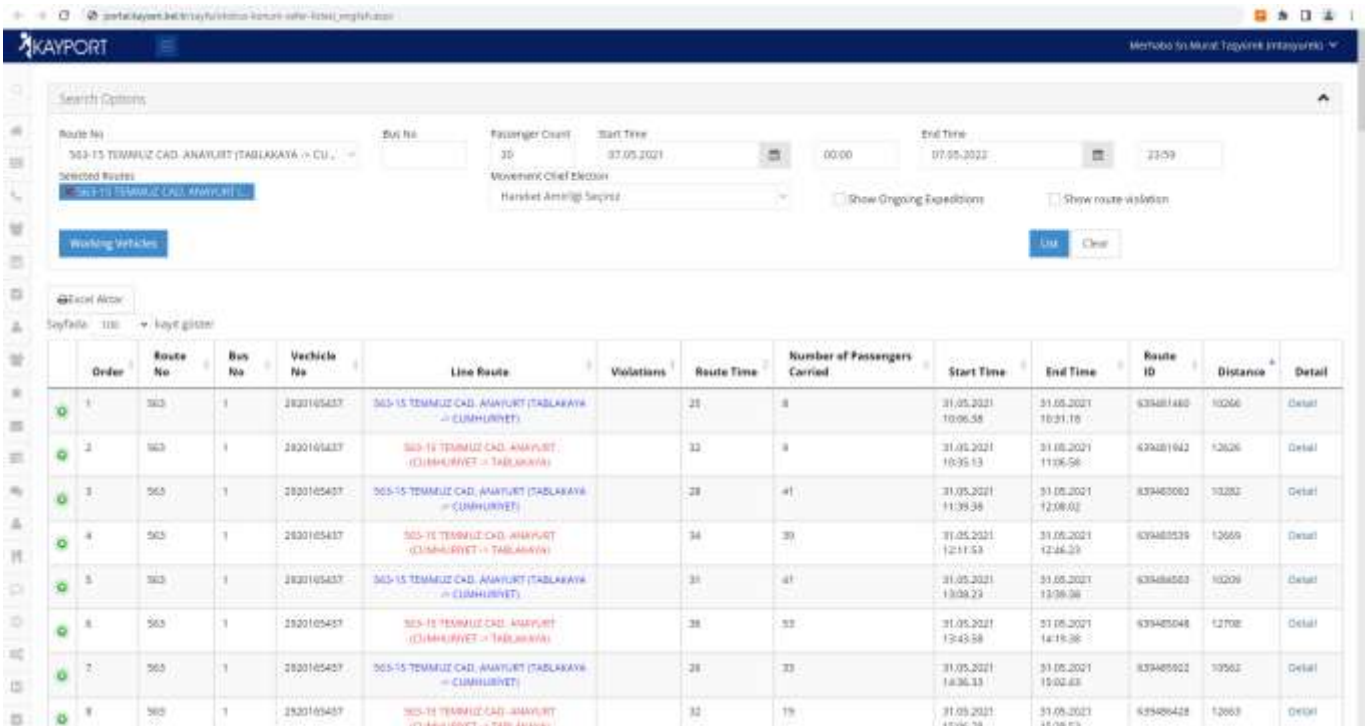


Fig. 7. Search Screen Used by KBB



Fig. 8. List of Violated Routes Used by KBB

REFERENCES

[1] P. K. Malik, R. Sharma, R. Singh, A. Gehlot, S. C. Satapathy, W. S. Alnumay, D. Pelusi, U. Ghosh, and J. Nayak, "Industrial internet of things and its applications in industry 4.0: State of the art," Computer Communications, vol. 166, pp. 125–139, 2021.

[2] V. Suma et al., "Internet-of-things (iot) based smart agriculture in indiaan overview," Journal of ISMAC, vol. 3, no. 01, pp. 1–15, 2021.

[3] M. Ghasemaghaci, "Understanding the impact of big data on firm performance: The necessity of conceptually differentiating among big data characteristics," International Journal of Information Management, vol. 57, p. 102055, 2021.

[4] C. Fan, D. Yan, F. Xiao, A. Li, J. An, and X. Kang, "Advanced data analytics for enhancing building performances: From data-driven to big

- data-driven approaches,” in *Building Simulation*, vol. 14, no. 1. Springer, 2021, pp. 3–24.
- [5] M. Naeem, T. Jamal, J. Diaz-Martinez, S. A. Butt, N. Montesano, M. I. Tariq, E. De-la Hoz-Franco, and E. De-La-Hoz-Valdiris, “Trends and future perspective challenges in big data,” in *Advances in Intelligent Data Analysis and Applications*. Springer, 2022, pp. 309–325.
- [6] J. Ranjan and C. Foropon, “Big data analytics in building the competitive intelligence of organizations,” *International Journal of Information Management*, vol. 56, p. 102231, 2021.
- [7] M. L. Larrea and D. K. Urribarri, “Visualization technique for comparison of time-based large data sets,” in *Conference on Cloud Computing, Big Data & Emerging Topics*. Springer, 2021, pp. 179–187.
- [8] J. D. Dinneen and C. Brauner, “Information-not-thing: further problems with and alternatives to the belief that information is physical,” 2017.
- [9] M. Vaitis, H. Feidas, P. Symeonidis, V. Kopsachilis, D. Dalaperas, N. Koukouroufli, D. Simos, and S. Taskaris, “Development of a spatial database and web-gis for the climate of greece,” *Earth Science Informatics*, vol. 12, no. 1, pp. 97–115, 2019.
- [10] M. Amin, G. W. Romney, P. Dey, and B. Sinha, “Teaching relational database normalization in an innovative way,” *Journal of Computing Sciences in Colleges*, vol. 35, no. 2, pp. 48–56, 2019.
- [11] S. Alqithami, “A serious-gamification blueprint towards a normalized attention,” *Brain Informatics*, vol. 8, no. 1, pp. 1–13, 2021.
- [12] I. Oditis, Z. Bicevska, J. Bicevskis, and G. Karnitis, “Implementation of nosql-based data wareh,” *Baltic Journal of Modern Computing*, vol. 6, no. 1, pp. 45–55, 2018.
- [13] I. Hrubaru, G. Talab’a, and M. Fotache, “A basic testbed for json data processing in sql data servers,” in *Proceedings of the 20th International Conference on Computer Systems and Technologies*, 2019, pp. 278–283.
- [14] Y. G. Chung, E. Haldoupis, B. J. Bucior, M. Haranczyk, S. Lee, H. Zhang, K. D. Vogiatzis, M. Milisavljevic, S. Ling, J. S. Camp et al., “Advances, updates, and analytics for the computation-ready, experimental metal-organic framework database: Core mof 2019,” *Journal of Chemical & Engineering Data*, vol. 64, no. 12, pp. 5985–5998, 2019.
- [15] P. Bouros and N. Mamoulis, “Spatial joins: what’s next?” *SIGSPATIAL Special*, vol. 11, no. 1, pp. 13–21, 2019.
- [16] V. K. Myalapalli, T. P. Totakura, and S. Geloth, “Augmenting database performance via sql tuning,” in *2015 International Conference on Energy Systems and Applications*. IEEE, 2015, pp. 13–18.
- [17] W. G. Pedrozo and M. S. M. G. Vaz, “A tool for automatic index selection in database management systems,” in *2014 International Symposium on Computer, Consumer and Control*. IEEE, 2014, pp. 1061–1064.
- [18] J. Correia, M. Y. Santos, C. Costa, and C. Andrade, “Fast online analytical processing for big data warehousing,” in *2018 International Conference on Intelligent Systems (IS)*. IEEE, 2018, pp. 435–442.
- [19] H. Sulistiani, S. Setiawansyah, and D. Darwis, “Penerapan metode agile untuk pengembangan online analytical processing (olap) pada data penjualan (studi kasus: Cv adilia lestari),” *Jurnal CoreIT: Jurnal Hasil Penelitian Ilmu Komputer dan Teknologi Informasi*, vol. 6, no. 1, pp. 50–56, 2020.
- [20] U. Erdinc, H. N. Bulus, and C. Erdoğan, “Veritabanı tasarımının yazılım performansına etkisi: Normalizasyona karşı denormalizasyon,” *Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü Dergisi*, vol. 22, no. 2, pp. 887–895, 2018.
- [21] B. Alshemaimri, R. Elmasri, T. Alsahfi, and M. Almotairi, “A survey of problematic database code fragments in software systems,” *Engineering Reports*, vol. 3, no. 10, p. e12441, 2021.
- [22] D. Milicev, “Hyper-relations: A model for denormalization of transactional relational databases,” *IEEE Transactions on Knowledge and Data Engineering*, 2021.
- [23] I. N. Chaparro-Cruz and J. A. Montoya-Zegarra, “Borde: Boundary and sub-region denormalization for semantic brain image synthesis,” in *2021 34th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. IEEE, 2021, pp. 81–88.
- [24] R. L. d. C. Costa, J. Moreira, P. Pintor, V. dos Santos, and S. Lifschitz, “A survey on data-driven performance tuning for big data analytics platforms,” *Big Data Research*, vol. 25, p. 100206, 2021.
- [25] A. H. Chill’on, D. S. Ruiz, and J. G. Molina, “Towards a taxonomy of schema changes for nosql databases: the orion language,” in *International Conference on Conceptual Modeling*. Springer, 2021, pp. 176–185.
- [26] E. Gupta, S. Sural, J. Vaidya, and V. Atluri, “Attribute-based access control for nosql databases,” in *Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy*, 2021, pp. 317–319.
- [27] J. Yang, Y. Yue, and K. Rashmi, “A large-scale analysis of hundreds of in-memory key-value cache clusters at twitter,” *ACM Transactions on Storage (TOS)*, vol. 17, no. 3, pp. 1–35, 2021.
- [28] A. Hillenbrand, U. St’orl, S. Nabiyeu, and M. Klettke, “Self-adapting data migration in the context of schema evolution in nosql databases,” *Distributed and Parallel Databases*, pp. 1–21, 2021.
- [29] A. Rafique, D. Van Landuyt, E. H. Beni, B. Lagaisse, and W. Joosen, “Cryptdice: Distributed data protection system for secure cloud data storage and computation,” *Information Systems*, vol. 96, p. 101671, 2021.
- [30] B. Vela, J. M. Caverro, P. C’aceres, A. Sierra, and C. E. Cuesta, “Defining a nosql document database of accessible transport routes,” in *2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, 2017, pp. 1125–1129.
- [31] S. P. R. Asaithambi, R. Venkatraman, and S. Venkatraman, “Mobda: Microservice-oriented big data architecture for smart city transport systems,” *Big Data and Cognitive Computing*, vol. 4, no. 3, p. 17, 2020.
- [32] M. T. Gonz’alez-Aparicio, M. Younas, J. Tuya, and R. Casado, “Testing of transactional services in nosql key-value databases,” *Future Generation Computer Systems*, vol. 80, pp. 384–399, 2018.
- [33] J. S. Fong, *Information Systems Reengineering, Integration and Normalization: Heterogeneous Database Connectivity*. Springer Nature, 2021.
- [34] M. Taşyürek, “Mekansal verilerin sıklıkla güncellendiği coğrafi bilgi sistemleri arama işleminde denormalizasyon yöntemi,” *Avrupa Bilim ve Teknoloji Dergisi*, no. 24, pp. 18–23, 2021.
- [35] J. Rand and A. Miranskyy, “On automatic parsing of log records,” in *2021 IEEE/ACM 43rd International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*. IEEE, 2021, pp. 41–45.

BIOGRAPHIES



MURAT TASYUREK was born in Turkey in 1986. He received the B.S. degree, the M.Sc. degree, and the Ph.D. degree in computer engineering from Erciyes University in 2009, 2011 and 2020, respectively. He worked as a software developer computer engineer for the Kayseri metropolitan municipality between 2010 and 2021. He has been working as an assistant prof in the computer engineering department of Kayseri University since 2021. His research interests include image registration, data mining, GIS (Geographic Information Systems) analysis, big data, deep learning, and optimization algorithms.