BŞEÜ Fen Bilimleri Dergisi
8(1), 472-483, 2021

BSEU Journal of Science
https://doi.org/10.35193/bseufbd.923267

e-ISSN:2458-7575 (https://dergipark.org.tr/tr/pub/bseufbd)

*Araştırma Makalesi - Research Article*

# Mondrian Based Real Time Anonymization Model

# Mondrian Tabanlı Kimliksizleştirme Modeli

İrem Civelek[1*], M. Ali Aydın[2]

***Geliş / Received: 20/04/2021*** ***Revize / Revised: 04/05/2021*** ***Kabul / Accepted: 04/05/2021***

**ABSTRACT**

The presence of private information belonging to individuals in data heaps called "Big Data" causes the privacy of the person to be endangered against disclosure attacks. To protect personal privacy in big data, it is ensured that anonymous data is created, stored, and shared in systems with anonymization methods. However, de-identified data cannot be reinstatement. The aim of this study is to create a new method that provides instant disidentification and does not disrupt the data structure in the system. In the study, the Hadoop ecosystem was used to process large data heaps. With the proposed model, it has been ensured that the requests from the user are processed in the Hadoop ecosystem with the services in the middle layer, thus obtaining anonymous data. The algorithm used for disidentification is optimized and results are compared according to algorithms in the literature. With the proposed model, it has been observed that the user is user-friendly in terms of querying and obtaining an anonymous data set. According to the analysis results, an algorithm that works with 40% efficiency compared to other algorithms in terms of processing speed was created with the disidentification algorithm used in the model.

***Keywords- De-Identification, Privacy Protection Model, Spark***

**ÖZ**

"Büyük Veri" olarak adlandırılan veri yığınlarında kişilere ait özel bilgilerin bulunması ifşa ataklarına karşı kişinin mahremiyetinin tehlikeye girmesine neden olmaktadır. Büyük veride kişi mahremiyetinin korunması için kimliksizleştirme yöntemleri ile kimliksiz veri oluşturup sistemlerde bu şekilde saklanması ve paylaşılması sağlanmaktadır. Fakat kimliksiz hale getirilen veride bilgi kaybı olduğu için veri eski haline döndürülememektedir. Bu çalışmanın amacı; büyük veri yığınları için anlık olarak kimliksizleştirme sağlayan ve sistemdeki veri yapısını bozmayan yeni bir yöntem oluşturmaktır. Çalışmada büyük veri yığınlarını işleyebilmek için Hadoop ekosistemi kullanılmıştır. Önerilen model ile kullanıcıdan gelen isteklerin ara katmanda bulunan servisler yardımı ile Hadoop ekosisteminde işlenmesi sağlanarak kimliksiz veri elde edilmesi sağlanmıştır. Kimliksizleştirme için kullanılan algoritma optimize edilerek kullanılmış ve literatürdeki algoritmalara göre avantajları kaydedilmiştir. Önerilen Modelle, kullanıcının sorgu çekmesi ve kimliksiz veri seti elde etmesi bakımından kullanıcı dostu olduğu görülmüştür. Analiz sonuçlarına göre, modelde kullanılan kimliksizleştirme algoritmasıyla işleme hızı bakımından diğer algoritmalara göre %40 verimli çalışan bir algoritma oluşturulmuştur.

***Anahtar Kelimeler- Kimliksizleştirme, Mahremiyet Koruma Modeli, Spark***

[1]*Sorumlu yazar iletişim: irem_civelek@hotmail.com (https://orcid.org/0000-0002-8995-7161)
*Bilgisayar Mühendisliği A.B.D, MSÜ Hezârfen Havacılık ve Uzay Teknolojileri Enstitüsü, Yeşilyurt, İstanbul*
[2]İletişim: aydinali@istanbul.edu.tr (https://orcid.org/0000-0002-1846-6090)
*Bilgisayar Mühendisliği A.B.D, MSÜ Hezârfen Havacılık ve Uzay Teknolojileri Enstitüsü, Yeşilyurt, İstanbul*
*Bilgisayar Mühendisliği Bölümü, İstanbul Üniversitesi-Cerrahpaşa, Avcılar, İstanbul*

BŞEÜ Fen Bilimleri Dergisi
8(1), 472-483, 2021

BSEU Journal of Science
https://doi.org/10.35193/bseufbd.923267

e-ISSN:2458-7575 (https://dergipark.org.tr/tr/pub/bseufbd)

## I. INTRODUCTION

Nowadays, wide variety and large numbers of data are emerging with the processes that produce digital data, the development of Internet of Things (IoT) devices, the use of social media, digitalization, and the management of all processes through programs, etc. In particular, IoT devices transfer the information they receive from sensors to databases continuously every millisecond. For this reason, it creates continuously increasing data size [1]. With the combination of these data, the concept of big data emerges, and problems arise regarding the storage and management of these data [2]. One of the problems in big data is the protection of personal privacy during the processing and storage of the collected data [3]. This protection is generally provided by methods, such as encryption, masking, and anonymization. Unlike other methods, the de-identification method provides data benefit because it does not break the data structure.

Direct sharing of relational data kept on my systems reveals three privacy threats: identity disclosure, membership disclosure, and sensitive attribute disclosure [4-6]. To eliminate these privacy threats, the identifiers that characterize the person are generalized by replacing them with the less specific value. However, as the size of the data grows, data consolidation, estimation, etc. privacy violations can be made with disclosure attacks. At this point, there is a need for systems and algorithms that can provide data security and process large amounts of data within an acceptable period. With the creation of the Hadoop system, big volumes of data can be processed with small and costless systems. Venugopal et al. [7] provide an improvement of up to 36 percent in processing speed by adapting the de-identification algorithm on the Hadoop environment. Jadhav et al. [8] also apply the same algorithm on a distributed basis to different data set sizes. However, as the data set grows, it cannot provide the same rate of performance increase. As the data grows, performance problems increase, Canbay [9] has proven that detecting and removing outlier data before de-identification before de-identification will increase processing speed to reduce the data set that will enter de-identification. However, in the studies conducted, there is a loss of information on the data after de-identification and data cannot be retrieved. In cases where the loss of information is not desired, a structure that will work with performance in different data set sizes is required for instant de-identification for each query transmitted to the system.

This paper has the following main contributions.

- We create a user-friendly system by anonymizing the data returned from the query.
- We provide that the data structure kept in the system is not damaged by not interfering with the data in main the system.
- We provide faster anonymization by using the memory calculation method of the Spark application.
- Our aim is to create a safe structure by choosing as a method that outlier data determine whether identity disclosure was made or not, and provide fast anonymization algorithm for real-time queries.
- We demonstrate that the performance of anonymization is increased with outlier data optimization.

In the rest of the article, sections are explained. General information about anonymization and big data technologies is mentioned in Section II. The related works on the subject are reviewed and the advantages of the model are introduced in Section III. The Mondrian-based real-time anonymization model is explained in Section IV. The results of the experiments are evaluated in Section V. Conclusions are present in Section VI.

## II. ANONYMIZATION AND BIG DATA TECHNOLOGIES

In this part of the article, the concepts of anonymization are discussed to better understand its methods. In addition, the advanced technologies used for processing data in large data sets and applying anonymization algorithms to data are mentioned.

### *A. Anonymization*

Anonymization is the replacement of attributes containing personal data with a more general value through generalization and suppression, permutation and anatomization methods [10]. In this way, the disclosure of attributes is prevented without spoiling the data structure. Table 1 shows examples of anonymized data.

**Table 1.** Identified Data and Anonymized Data

| **Identified Data** | | | **Anonymized Data** | | |
|---|---|---|---|---|---|
| **Age** | **Gender** | **Illness** | **Age** | **Gender** | **Illness** |
| 24 | Female | HIV | 20-30 | Person | HIV |
| 30 | Male | Ulcer | 20-30 | Person | Ulcer |
| 32 | Female | Asthma | 30-40 | Person | Asthma |

Attributes must be classified to anonymization. These attributes are explained below [11].

• Explicit Identifier: Attributes that directly reflect the person. Name, Surname, ID, etc.

• Quasi Identifier: These are attributes that, even if they do not directly reflect a person, have the potential to reflect a particular person when combined with other attributes. Age, Gender, Profession, etc.

• Sensitive attribute: These are the sensitive attributes belonging to the person who will cause problems in case of being handled by third parties. Disease, Disability situation, etc.

• Non-Sensitive attribute: These are non-sensitive attributes that do not cause any problems if they are captured by third parties.

For anonymization algorithms to be viable and prevent disclosure attacks, they must meet the k-anonymity, l-diversity, t-proximity criteria.

• K-anonymity [4]: It means that there are k-1 of the same value in each of the combinations formed by combining the features in the data set.

• L-diversity [6]: For the same attribute combinations in the data set, the sensitive attribute must have at least L diversity.

• T-closeness [12]: In the case that each of the features in the data set is subclassified according to their proximity to each other, the T-proximity state must be provided.

Since the data set given in Table 2 does not meet the above criteria, it causes sensitive attribute disclosure of the person known to be a male and an engineer in the data set according to the values in the 3rd and 4th rows.

**Table 2.** Anonymized Data Set

| **Order** | **Age** | **Gender** | **Profession** | **Illness** |
|---|---|---|---|---|
| **1** | 40-50 | Female | Lawyer | HIV |
| **2** | 40-50 | Female | Lawyer | Ulcer |
| **3** | 40-50 | Male | Engineer | Hepatitis |
| **4** | 40-50 | Male | Engineer | Hepatitis |
| **5** | 30-40 | Female | Doctor | Ulcer |
| **6** | 30-40 | Female | Doctor | Asthma |

### *B. Outlier Data*

For the data to be generalized, it must first be divided into certain clusters. When data is classified according to attributes, data that cannot be included in any cluster are called outlier data. Data that cannot belong to A and B clusters in Figure 1 are outliers. Since generalization cannot be provided for such data, it creates a weakness for anonymization algorithms [13].

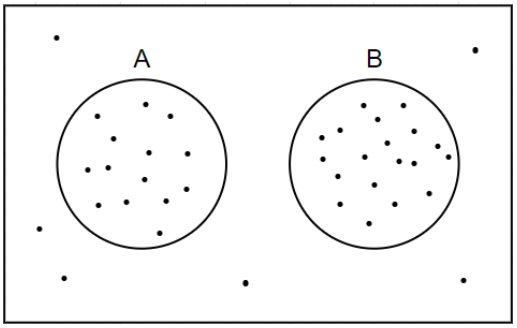



**Figure 1.** Clustered Data

### *C. Apache Hadoop*

Along with the concept of big data, this size of data processing, storage, etc. problems are starting to occur. Hadoop is an ecosystem that overcome these problems, providing the opportunity to store and process data in cheaper and smaller systems instead of large and costly systems [14]. It brings together many idle and low-performance computers in its ecosystem, allowing them to be stored and processed in different computers with the HDFS [15] (Hadoop Distributed File System) file system.

There is a Master node and many Slave nodes in the Hadoop architecture. Job Tracker in the master node ensures that the transactions are transmitted to the nodes and coordinated. There is a Data Node in each Slave node that enables its data to be stored. With the HDFS file system, this data behaves like a single database. To distribute the tasks to be done to Slave Nodes, it uses the Map-Reduce [16] method, which was first used by Google. There are 2 functions in this method, Map and Reduce. The map function divides the process into smaller pieces and distributes it to other worker nodes. After the worker nodes perform the given task in parallel, it sends the result to the master node. The results returned from the worker node are combined with the Reduce function. Hadoop architecture is shown in the Figure 2.

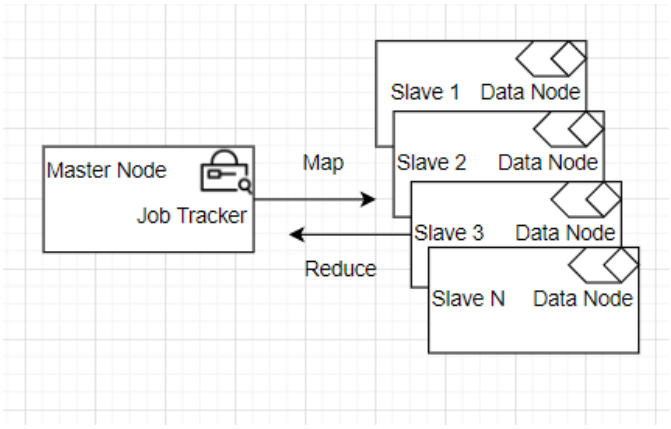



**Figure 2.** Hadoop Architecture

Projects are developed to solve performance problems on the Hadoop ecosystem, such as data access or processing speed. Cloudera Impala and Spark are projects developed as a solution to these problems.

*1) Cloudera Impala:* HIVE is used within the Hadoop ecosystem to access data via HDFS. HIVE derives SQL queries from Slave nodes by converting them to the Map-Reduce method [17]. However, HIVE does not respond quickly enough to real-time queries. The Impala project, developed by Cloudera as a solution to this problem, is an open-source distributed SQL technology that provides low latency, high-performance data queries in the Hadoop environment [18]. To obtain data over HDFS, instead of the Map-Reduce method, it accesses data directly with distributed queries. In this way, Cloudera impala runs 6 to 55 times faster than HIVE [19].

*2) Apache Spark:* Apache Spark is a technology developed on the Hadoop flexible architecture to process big data in a fast and scalable way. It is claimed to work 100 times faster than HDFS and Map-Reduce methods [20]. It does in-memory computation by characterizing distributed clusters and thus obtains much faster results.

According to the Spark architecture, Tasks (T1, T2, .., TN) that will work in each cluster are created and these Tasks are run in parallel with the Executer. Cluster Manager, on the other hand, ensures the coordination and operation of the components. Spark architecture is shown in Figure 3.

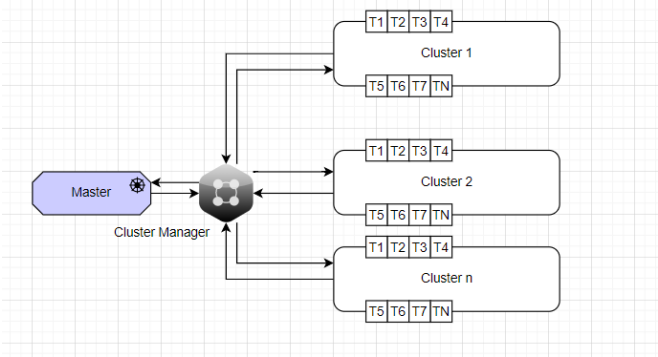



**Figure 3.** Spark Architecture

## III. RELATED WORK

In the generalization methods used in the anonymization algorithms, the frequency value of the data was first used by Sweeney [21] to create a greedy generalization algorithm. K-anonymity is provided by the fact that the frequency values of the identifiers formed are lower than k. However, this algorithm is insufficient to provide minimum generalization. Sweeney [22] created an algorithm that gives many minimal generalizations based on an information metric called Precision. It prefers generalization with minimal distortion from generalizations created by creating many generalizations. Since it tries all the generalizations one by one, it does not work well in medium-sized data. Wang et al. [23] proposed an algorithm that provides generalization from the bottom upwards. According to this algorithm, the rate of distortion in the data increases according to the number of the hierarchy level. Therefore, it provides a local optimum solution instead of a general optimum solution. Fung et al. [24] try to achieve optimum k-anonymity by performing top-down generalization (TDS). However, the TDS algorithm works slowly compared to other algorithms, since it is tried and created sequential sets for each feature. LeFevre et al. [25], unlike other algorithms, proposed the Mondrian algorithm, which makes multi-dimensional generalization rather than one-dimensional generalization. Mondrian uses a multidimensional re-encoding method using a greedy algorithm to find an equivalence class. In this way, it gets faster results compared to one-dimensional generalization algorithms.

The presence of outlier data in the data set to be created creates identity disclosure risk. Therefore, Wang et al. [26] while detecting and removing the anomalous data in anonymization in the anonymization algorithm they developed, Majeed et al. [27], on the other hand, provide data benefit by including it again in the data structure. However, this creates an extra cost for the algorithm. Canbay [28], in his study with the Mondrian algorithm to reduce this cost, ensures that the outlier data is detected and removed from the data before anonymization. In this way, A more performance structure is obtained with the Mondrian algorithm. Canbay is not testing its algorithm for the Spark environment in this study.

Torticar [29], Ashkouti et al. [30] prove that when they adapt the Mondrian algorithm on Spark, they get faster results compared to other studies. However, in these studies, no optimization is carried out for outlier data. In the proposed model, the modified de-identification algorithm, which enables the data set to be optimized for outlier data is implemented in Spark. Accordingly, a real-time anonymization model, which works faster than other studies, is created. Unlike other studies, in this model, while the data is re-anonymized and presented to the client in each query request, no changes are made to the data format stored in the system. In this way, it is aimed to protect the data structure kept in the system.

**Table 3.** Comparison of generalization algorithms

| *Algorithm* | *Hierarchy* | *Dimension* | *Minimum Generalization* | *Searching Algorithm* | *Generalization* |
|---|---|---|---|---|---|
| Datafly [21] | Bottom Up | Single | No | Complete | Global |
| MinGen [22] | Bottom Up | Single | Yes | Not feasible | Global |
| Bottom-up Generalization [23] | Bottom Up | Single | Yes | Greedy | Global |
| TDS[24] | Top Down | Single | Yes | Greedy | Local |
| Mondrian [25] | Top Down | Multi | Yes | Greedy | Local |

## IV. MONDRIAN BASED REAL TIME ANONYMIZATION MODEL

The purpose of the model is to provide communication between the Hadoop environment and the relational database users. By adapting the queries from the client in SQL format to the data stored in the Hadoop environment, the obtained data is instantly anonymous and presented to the client. In this way, it is ensured that the client can obtain the data he wants without an identity without the adaptation process to the big data environment.

In the system to be installed, the query results transmitted to the Hadoop environment must pass certain controls to meet the security criteria. To prevent disclosure attacks with the obtained data set, there should be no outlier data in the data set. Therefore, it was ensured that data contradiction was detected before starting the anonymization process. When the data is made ready for anonymization, if the anonymization criteria are not provided, the forwarded query is considered a risky query.

There are 3 basic sections in the model to establish a connection between the client and the Hadoop environment; to parse out the outlier data, to prevent the returning query result from creating identity disclosure, and to perform instant anonymization. The first part provides to obtain data from the Hadoop environment with SQL queries, the second part prepares the data and applies the security criteria, and the third part presents the problem-free data set to the client. The architecture of the proposed model is shown in Figure 4.

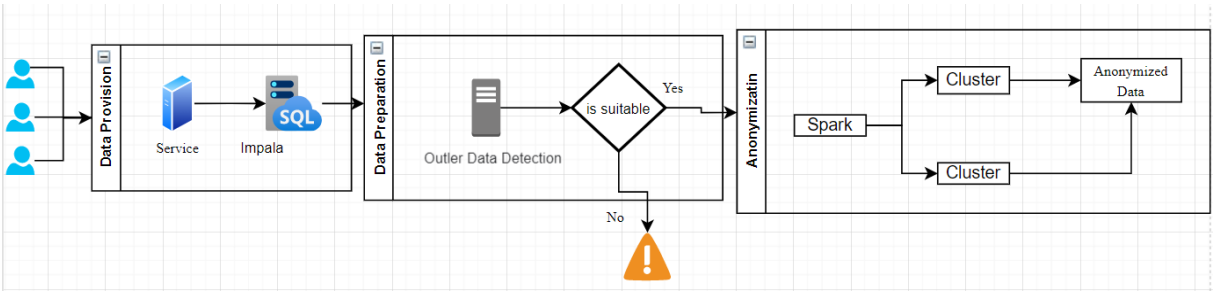



**Figure 4.** Mondrian Based Real-Time Anonymization Model

### *A. Data Provision Layer*

In the proposed model, the data request of the system is provided directly by the users. SQL queries from the user cannot provide data directly from the Hadoop environment. At this point, a layer is needed to provide communication between the Hadoop system and the clients. Communication between the Hadoop system and the clients is provided at the data provision layer in between. SQL queries transmitted to the system are captured by the service in this layer. Using Impala libraries, data is obtained in a distributed manner in the Hadoop environment. The data obtained is transferred to the second layer to be prepared for the anonymization process. Figure 5 shows the data sets returned from Impala.

| | age | workclass | fnlwgt | education | educationnum | maritalstatus | occupation | relationship | race |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 39 | State-gov | 77516 | Bachelors | 13 | Never-married | Adm-clerical | Not-in-family | White |
| 2 | 50 | Self-emp-not-inc | 83311 | Bachelors | 13 | Married-civ-spouse | Exec-managerial | Husband | White |
| 3 | 38 | Private | 215646 | HS-grad | 9 | Divorced | Handlers-cleaners | Not-in-family | White |
| 4 | 53 | Private | 234721 | 11th | 7 | Married-civ-spouse | Handlers-cleaners | Husband | Black |
| 5 | 28 | Private | 338409 | Bachelors | 13 | Married-civ-spouse | Prof-specialty | Wife | Black |
| 6 | 37 | Private | 284582 | Masters | 14 | Married-civ-spouse | Exec-managerial | Wife | White |
| 7 | 49 | Private | 160187 | 9th | 5 | Married-spouse-absent | Other-service | Not-in-family | Black |
| 8 | 52 | Self-emp-not-inc | 209642 | HS-grad | 9 | Married-civ-spouse | Exec-managerial | Husband | White |

**Figure 5.** Impala Data Set

The data sets obtained from this layer are transmitted to the data preparation layer, which is the preprocessing layer, to be extracted from the outliers.

***B. Data Preparation Layer***

The second layer acts as a provision for the anonymization layer, while at the same time protecting against the system's disclosure attacks. Outlier detection and alarm mechanism are in this layer.

Clustering algorithms used in machine learning methods are used to detect outlier data. WEKA program is software written in the java programming language that includes clustering algorithms and can be used by converting libraries into jar files [31]. The data set is clustered with the Expectation-Maximization (EM) [32] algorithm in the WEKA program according to its specified attributes. EM algorithm obtains data sets containing the maximum amount of data with predictive criteria.

Clusters determined according to the specified attributes must have a certain size. Clusters that do not have a predetermined threshold value according to system features and security level are considered as outlier data and extracted from the data. Figure 6 shows the number of records in each cluster as a result of the sample cluster. Accordingly, the third cluster was considered as outliers and separated from the data because it did not have sufficient size.

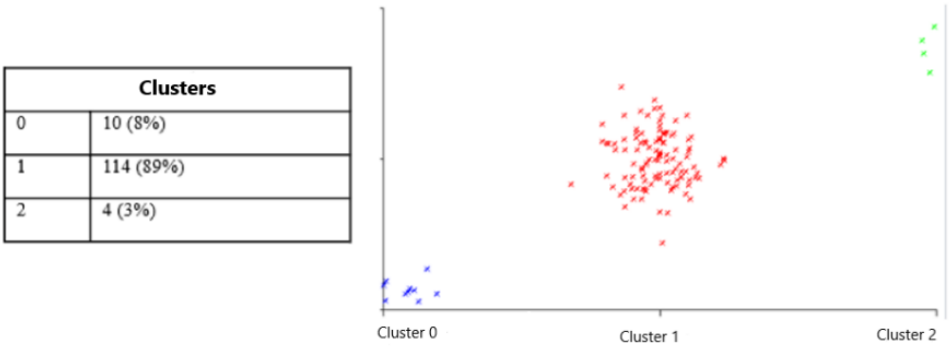



**Figure 6.** Clusters

After the outliers are cleared from the data set, it is checked whether the data set meets the de-identification criteria. At this stage, if the data set does not have a certain size and does not meet the de-identification criteria, the query sent to the system is evaluated as a query that can allow identity disclosure, and system logging is performed. The recorded query is reported and forwarded to the relevant units.

The dataset obtained from this layer, cleaned from outlier data and ready to be processed, is transferred to the third layer, the anonymization layer.

### *C. Anonymization Layer*

The data that is cleared from the data contrary to the third layer, the anonymization section, is transferred ready for processing. Through Spark, this data is processed on more than one cluster in parallel. Functions given in Algorithm 1 are transferred to Clusters via Master Node.

The Mondrian-based algorithm is used for anonymization in the system. Mondrian code is given in Pseudocode 1.

**PSEUDOCODE 1:** MONDRIAN ANONYMIZATION PSEUDOCODE

```
Input:
        S: Dataset
Output:
        V: Anonymized dataset
Start:
        Divide S into T clusters

        For each cluster:

                fs ← claculate_frequency (S, dim)
                splitV al ← find_median (fs)
                lhs ← {t ∈ S: t.dim ≤ splitV al}
                rhs ← {t ∈ S: t.dim > splitV al}
                V ← Anonymize ()

        Return V
```

**Figure 7.** Mondrian Anonymization Pseudocode

### *D. Summary*

The Mondrian-based real-time anonymization algorithm returns an input transmitted in SQL syntax, V anonymous data set. In Pseudocode 2, the pseudo-code of the Mondrian-based anonymization algorithm is explained.

**PSEUDOCODE 2:** MONDRIAN BASED REAL TIME ANONYMIZATION PSEUDOCODE

```
Input:
        S: SQL Query
Output:
        V: Anonymized Dataset
Start:
        S ←SQL query
        Dataset ← Impala(S)
        Clustering (Dataset)

        Loop: Cluster Count

                If: The amount of data of the cluster is less than the threshold value
                        Detect outlier data.
                        Remove from Dataset.

        End Loop:

        If: Dataset is not suitable for anonymization

                Return: null
                Alert: disclosure
        Else:
                Spark: Send Dataset to clusters
                        Start Process on clusters
                        V ← Mondrian (Dataset)

        Return: V
```

**Figure 8.** Mondrian Based Real Time Anonymization Pseudocode

The input of the model function is SQL statements from the user. The data set is obtained by distributing SQL queries to the HDFS system with Impala libraries. The obtained data set is divided into T clusters after clustering. Each cluster must have more than the threshold value. Data that cannot be included in sets are purged from the data set. The data set suitable for anonymization is de-identified by distributing it to the clusters on the Spark library. The general flow diagram is given in Figure 9.

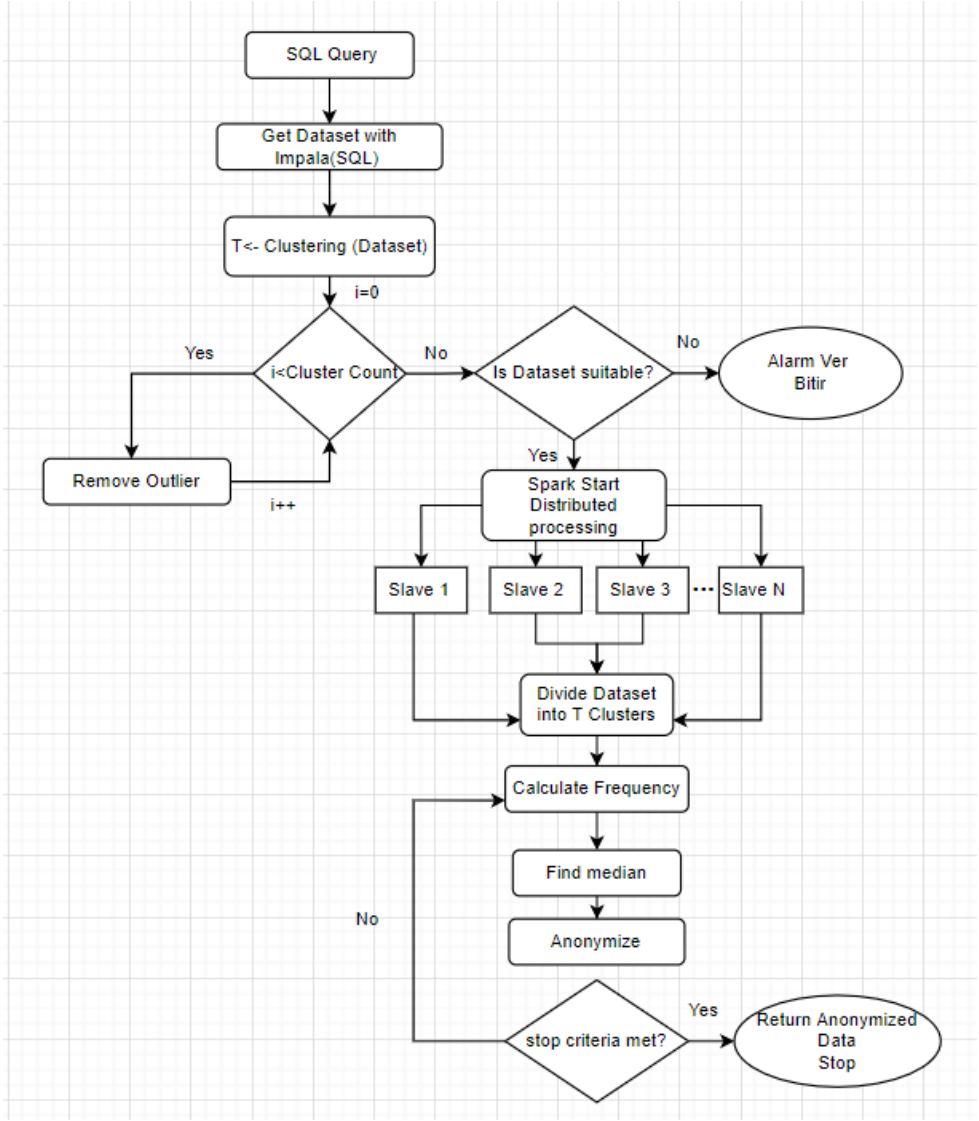



**Figure 9.** Mondrian Based Real Time Anonymization Model Diagram

## V. RESULTS

In the testing of the model, the Adult data set, which is frequently used in the literature, was used. The Adult dataset was created by Barry Becker in the 1994 Census [33]. There are a total of 15 fields in this data set. It contains 6 semi-descriptive fields used for classification. The model was tested on a 64bit, i7 core processor, 4-core machine. Figure 10 lists the results according to k = 2, k = 5 and k = 10 anonymity criteria.

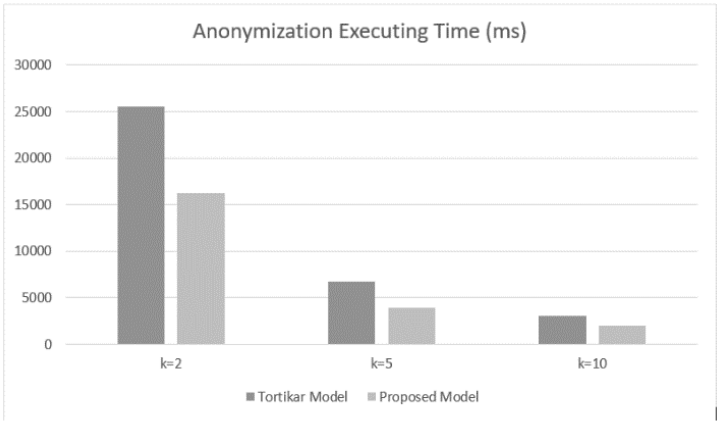



**Figure 10.** Comparison of the Tortikar [29] Model and the Proposed Model

According to the results, anonymization after removing the outliers from the system according to different anonymity criteria increases the speed of the algorithm. In the study, it is seen that about 40% improvement was made in terms of the rate of anonymization in different anonymity criteria, such as k = 2, k = 5, and k = 10. Figure 11 shows sample data obtained as a result of anonymization.

| sensitive_data | education_num | age | capital_gain | hours_per_week | fnlwgt |
|---|---|---|---|---|---|
| [<=50K] | [9,10]\|[30,19] | [0,5013] | [40,58]\|[289980,101509] |
| [<=50K] | [9,10]\|[30,19] | [0,5013] | [40,58]\|[289980,101509] |
| [<=50K] | [9,10]\|[30,19] | [0,5013] | [40,58]\|[289980,101509] |
| [<=50K] | [9,10]\|[30,19] | [0,5013] | [40,58]\|[289980,101509] |
| [<=50K] | [9,10]\|[30,19] | [0,5013] | [40,58]\|[289980,101509] |
| [<=50K] | [9,10]\|[30,19] | [0,5013] | [40,58]\|[289980,101509] |
| [>50K] | [9,10]\|[30,19] | [0,5013] | [40,58]\|[289980,101509] |
| [<=50K] | [9,10]\|[30,19] | [0,5013] | [40,58]\|[289980,101509] |
| [<=50K] | [9,10]\|[30,19] | [0,5013] | [40,58]\|[289980,101509] |
| [<=50K] | [9,10]\|[30,19] | [0,5013] | [40,58]\|[289980,101509] |
| [<=50K] | [9,10]\|[30,19] | [0,5013] | [40,58]\|[289980,101509] |

**Figure 11.** Anonymized Data

## VI. CONCLUSIONS

It seems that the proposed model is a user-friendly model. The purpose of the study is to return a quick response to the client. Unlike HDFS and Map-Reduce methods, using Spark makes the model more powerful. In addition, it is determined that by detecting outlier data in advance and removing it from the data, 40% faster results are obtained compared to Mondrian applications in the literature.

In this study, a new utility-based anonymization model has been proposed that provides real-time anonymization in big data, which increases the performance to be obtained from the anomaly data and aims to make the calculation cost-effective. In the proposed model, the total speed was increased by removing the outlier data. The results obtained from the experiments show that the proposed model is successful in increasing the processing speed for anonymization and ensuring anonymity.

## KAYNAKLAR

[1] Erdoğan, H., Küçük, K. & Khan, S. A. Endüstriyel IoT Bulut Uygulamaları için Düşük Maliyetli Modbus/MQTT Ağ Geçidi Tasarımı ve Gerçekleştirilmesi. *Bilecik Şeyh Edebali Üniversitesi Fen Bilimleri Dergisi*, *7*(1), 170-183.

[2] Nasser, T. & Tariq, R. S. (2015). Big data challenges. *J Comput Eng Inf Technol 4: 3. doi: http://dx. doi. org/10.4172/2324*, *9307*(2).

[3] Mehmood, A., Natgunanathan, I., Xiang, Y., Hua, G. & Guo, S. (2016). Protection of big data privacy. *IEEE access*, 4, 1821-1834.

[4] Sweeney, L. (2002). k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05), 557-570.
[5] Nergiz, M. E., Atzori, M. & Clifton, C. (2007). Hiding the presence of individuals from shared databases. In Proceedings of the 2007 ACM SIGMOD international conference on Management of data, 665-676..
[6] Machanavajjhala, A., Kifer, D., Gehrke, J. & Venkitasubramaniam, M. (2007). l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1), 3-es.
[7] Venugopal, V. & Vigila, S. M. C. (2018). Implementing Big Data Privacy with MapReduce for Multidimensional Sensitive Data. *International Journal of Applied Engineering Research*, 13(15), 11824-11829.
[8] Jadhav, R. H. (2018). Distributed Bottom up Approach for Data Anonymization using Map Reduce framework on Cloud. *Internatıonal Journal*, *3*(6).
[9] Canbay, Y., Vural, Y. & Sağıroğlu, S. (2018). Privacy preserving big data publishing. In 2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT). 24-29
[10] Goswami, P. & Madan, S. (2017). A survey on big data & privacy preserving publishing techniques. *Advances in Computational Sciences and Technology*, 10(3), 395-408.
[11] Wang, L., Jajodia, S. & Wijesekera, D. (2004). Securing OLAP data cubes against privacy breaches. In IEEE Symposium on Security and Privacy, 2004, 161-175.
[12] Li, N., Li, T. & Venkatasubramanian, S. (2007). t-closeness: Privacy beyond k-anonymity and l-diversity. In 2007 IEEE 23rd International Conference on Data Engineering, 106-115.
[13] Kohlmayer, F., Prasser, F. & Kuhn, K. A. (2015). The cost of quality: Implementing generalization and suppression for anonymizing biomedical data with minimal information loss. *Journal of biomedical informatics*, 58, 37-48.
[14] Apache Hadoop. (2006). The Apache Software Foundation, https://hadoop.apache.org/ (25.03.2021)
[15] Shvachko, K., Kuang, H., Radia, S. & Chansler, R. (2010). The hadoop distributed file system. In 2010 IEEE 26th symposium on mass storage systems and technologies (MSST), 1-10.
[16] Dean, J. & Ghemawat, S. (2004). MapReduce: Simplified data processing on large clusters. 6th Symposium on Operating Systems Design and Implementation, 137-149
[17] Apache Hive. (2011). Apache Hive TM, https://hive.apache.org/ (18.04.2021).
[18] Apache Impala. (2021). Apachecon, https://impala.apache.org/overview.html (18.04.2021).
[19] Kornacker, M., Behm, A., Bittorf, V., Bobrovytsky, T., Ching, C., Choi, A., ... & Yoder, M. (2015). Impala: A Modern, *Open-Source SQL Engine for Hadoop. In Cidr*, 1, 9.
[20] Spark Apache. (2011). The Apache Software Foundation, http://spark.apache.org/ (26.03.2021).
[21] Sweeney, L. (1998). Data fly: A system for providing anonymity in medical data. *In Database Security XI*, 356-381.
[22] Sweeney, L. (2002). Achieving k-anonymity privacy protection using generalization and suppression. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05), 571-588.
[23] Wang, K., Yu, P. S. & Chakraborty, S. (2004,). Bottom-up generalization: A data mining solution to privacy protection. In Fourth IEEE International Conference on Data Mining (ICDM'04), 249-256.
[24] Fung, B. C., Wang, K. & Yu, P. S. (2005). Top-down specialization for information and privacy preservation. In 21st international conference on data engineering (ICDE'05), 205-216.
[25] LeFevre, K., DeWitt, D. J. & Ramakrishnan, R. (2006). Mondrian multidimensional k-anonymity. In 22nd International conference on data engineering (ICDE'06), 25-25.
[26] Wang, H. & Liu, R. (2009). Hiding distinguished ones into crowd: privacy-preserving publishing data with outliers. In Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, 624-635.
[27] Majeed, A. (2019). Attribute-centric anonymization scheme for improving user privacy and utility of publishing e-health data. *Journal of King Saud University-Computer and Information Sciences*, 31(4), 426-435.
[28] Canbay, Y., Vural, Y. & Sağıroğlu, Ş. (2020). OAN: outlier record-oriented utility-based privacy preserving model. *Journal of the Faculty of Engineering and Architecture of Gazi University*, 35(1), 355-368.
[29] Tortikar, P. (2019). K-Anonymization Implementation Using Apache Spark, Master of Science, North Dakota State University, Department of Computer Science, Fargo, North Dakota.
[30] Ashkouti, F. & Sheikhahmadi, A. (2021). DI-Mondrian: Distributed improved Mondrian for satisfaction of the L-diversity privacy model using Apache Spark. *Information Sciences*, 546, 1-24.

[31] Gündüz, H. (2020). WEKA Veri Madenciliği Yazılımının Sürümleri Arasındaki Kalite Değişimlerinin QMOOD ile İncelenmesi. *Bilecik Şeyh Edebali Üniversitesi Fen Bilimleri Dergisi*, 7(2), 825-836.

[32] Sezgin, E. & Çelik, Y. (2013). Veri madenciliğinde kayıp veriler için kullanılan yöntemlerin karşılaştırılması. Akademik Bilişim Konferansı, Akdeniz Üniversitesi, 23-25.

[33] Adult Data Set. (1994). The UCI Machine Learning Repository, https://archive.ics.uci.edu/ml/datasets/Adult (26.03.2021).