International Journal of Safety and Security Engineering
Vol. 15, No. 7, July, 2025, pp. 1361-1368
Journal homepage: http://iieta.org/journals/ijsse

Towards a Security Model for Dynamic Access Control in Graph Databases

Samira Telghamti*, Lakhdar Derdouri, Abdelhabib Bourouis

Rela(cs)2 Laboratory, Math and Computer Science Department, Larbi Ben M'hidi University, Oum El Bouaghi, P.O. Box 358, Algeria

Corresponding Author Email: samira.telghamti@univ-constantine2.dz

https://doi.org/10.18280/ijsse.150703

Received: 3 June 2025
Revised: 10 July 2025
Accepted: 20 July 2025
Available online: 31 July 2025

Keywords:
Big Data, dynamic role-based access control, graph database security, NoSQL database

ABSTRACT

Since databases were created and distributed, privacy and security have gained a lot of attention in the computer community. With the advancement of networks and the arrival of Cloud computing, most of the databases are publicly and remotely accessible, so they become exposed to several kinds of security threats. However, as far as we know, the existing access control security models for NoSQL databases are static, meaning the access policy remains unchanged for a given user, and does not consider his behavior during the utilization of the database. This paper presents an innovative dynamic access control model specifically designed for graph databases; the model dynamically adapts user roles based on user interactions and monitors unexpected behavior to enhance security. To handle the dynamicity aspect, we have opted for the trust concept, where we compute a given user's reputation degree according to the role he has been assigned. For every user, a trust level is computed and continually updated, according to the actions that the user performs. Based on the current trust level, the roles of the user are changed. Experimental results based on realistic scenarios show that the proposed model allows to dynamically update user roles, thus guaranteeing the security of the database.

1. INTRODUCTION

Relational databases were the most used and studied for several decades [1, 2]. These databases amply met the needs of information systems that were being designed several years before. Indeed, factual data, often textual in nature, are highly structured, providing them with well-formalized access and query models, particularly via relational database management systems DBMS [3, 4], and structured query language (SQL) [5, 6]. Due to their structured data and their access policies, security issues and protection procedures for SQL databases are successfully formulated and implemented. However, the advent of NoSQL databases that emerged as an alternative solution for data and knowledge storage aiming to ensure scalability and availability that are very suitable for modern data storage, namely for Big Data [7, 8], has imposed new approaches for database security and privacy. Indeed, these new databases resolve several issues, such as high volume of data, data variability, and data volatility. Also, and among several advantages of NoSQL databases, is their capacity for effectively processing and storing unstructured data, including semi-structured data from social networks, texts, and graphics [9, 10]. However, and despite that NoSQL databases can deal with wide volumes of personnel and sensitive data, up to now, such databases cannot be effectively protected by using classical techniques, designed for relational databases [11, 12]. Such a fact is mainly due to the high variability of stored data that can include active content and to the high level of openness in such databases and freedom that should be accorded to the users to guarantee high operational efficiency. Moreover, and in addition to conventional vulnerabilities associated to databases, such as SQL injection attacks, where authors continue to propose efficient solutions, even for NoSQL databases [13], access control in databases is one of the most important techniques which help to protect data and ensure confidentiality, including user's privacy and database security [1, 14]. In order to successfully prevent and halt attacks and handle them when they happen, the database's access control model must offer a high level of security. The latter part of database security is crucial, especially when the databases are going to be openly and remotely accessed, such as in cloud computing and web applications. Our study presents a novel dynamic model, to deal with NoSQL database security and privacy. To design the security model, the proposed method proceeds by natively integrating the security metadata into graph-oriented databases. Based on the issues related to the traditional access control paradigm, this new model seeks to:

- Establish a trust-based dynamic control to access the graph databases: the access control is based on trust and is distinguished by its dynamic aspect, in contrast to the classical access control models, which are static.
- The enhancement of the access restriction in graph-oriented databases: This objective is guaranteed by considering the user's past behavior. Consequently, a user's rights and permissions may alter based on his actions when accessing the database and using it. It should be noted that the primary objective of our work is to offer designers the

best guidelines to implement dynamic access control mechanisms for graph-oriented databases.

The remainder of the paper is organized as follows: Section 2 reviews some recent related work from the literature. Section 3 is devoted to the proposed access control model in graph databases. Section 4 exposed the results of experimentation and their discussion. Finally, in Section 5, we conclude our paper and highlight some future perspectives.

2. RELATED WORK

In informatics, access control to network resources, including local and remote databases, is an important and critical issue for both administrators and users. This topic has gained even more relevance in recent years due to the exponential growth of data and the increasing complexity of IT infrastructures. Since databases first came into existence, access control has been widely discussed, and several models were proposed for classical databases—especially relational ones, given their dominance over the past half-century. These traditional models, however, were not designed to address the unique challenges posed by modern non-relational systems. First of all, it should be noted that access control to NoSQL databases strongly depends on the NoSQL database model itself, as each type offers distinct structural and operational characteristics. We identify four families of NoSQL database models, each catering to specific data storage and retrieval needs in contemporary applications.

1) Associative model, where data is stored in the form of keys, usually hash codes, and the data values corresponding to these keys [15]. Redis and Oracles databases are examples of databases that are based on the associative model.

2) Column-based model, where keys in this case are obtained by combining several columns, rows and time stamps [16]. It has been reported by several authors that this model allows faster querying of the database [17].

3) Document-based model, where data is stored as documents and where access to them is provided by a set of keys. This type of model supports different types of data, ranging from structured data to unstructured data (Text) and passing by semi-structured data (XML) [16].

4) Graph-based models, where data is stored in a graph structure, involving vertices and edges [18]. This model is well suited to social networks, recommendation applications. However, the model is known for its relatively high complexity compared to the other models [19].

SQL databases are appropriate to integrate traditional access control mechanisms due to their structural characteristics [20]. In NoSQL databases, only few works have integrated native access control [21]. As a case of access control model in NoSQL databases, a security model for a Graph-Oriented database has been proposed in reference [22], wherein the Management System incorporates the model. To do that, they used metadata with authorization rules to control access in applications that use a graph database. However, they only consider protection at the vertex level. Although authors have dealt with the dynamic aspect of NoSQL Database (Neo4j), the proposed role-based mechanism cannot be considered as dynamic. In reference [23], a general framework has been proposed for building document-based databases with the incorporation of security mechanisms as native aspects of the database. The work is based on a novel approach for enhancing the Mongo DB RBAC purpose-based access control model. The authors have focused and have been limited to document-oriented NoSQL databases and without considering user behavior and trust in their work. In reference [24], the access control paradigm allows access control policies to be specified and enforced at various resource hierarchy levels, including a column, row, or column family. The latter was designed to operate with HBase and Cassandra, where authors focus on the different access policies at every level of resource hierarchy, rather than focusing on user behavior. So, the dynamicity aspect of the proposed model was not addressed and resolved. Finally, in reference [25], the authors describe the unified framework and formal language ReLOG for encoding ReBAC rules, which are basically graph queries. In this work, role-based access control and dynamicity concepts were not addressed. Ahmadi and Derek [26] have proposed an Attribute Based Access Control (ABAC) model and its implementation on a graph database. They experimented their model with a sample use-case, where they were able to evaluate several access policies according to paths in the graph representation. In this work, dynamicity is ensured by policy matching at runtime. However, the user behavior is not addressed. Recently, data and machine-learning techniques start to be used for dynamical access control in advanced databases, including graph ones. Magomedov et al. [27] have dealt with anomaly detection in graph databases, aiming at detecting frauds, by mining sub-graph patterns using a machine learning algorithm, the access control can be considered as dynamic; however, it is not role-based and has not considered the user behavior. Several authors continue to propose novel secure design and implementation of graph-based databases [28]. At design level, Paneque et al. [29] introduced in their paper a knowledge based framework for secured graph-based databases, assuming that the most proposed frameworks consider security issues in such databases at implementation level. The authors of this work used ontologies to simultaneously take into account database modeling and security requirements. However, they have not discussed the dynamicity aspect of their framework. A role-based access control model for graph databases was proposed by Chabin et al. [30], which supports schema constraints and constraint rules aiming to protect data. Authors claim that the proposed model allows rewriting, planning and executing queries in parallel while respecting the access constraints. Like previous reviewed work, the latter cannot be considered as dynamic, given that the constraints are predefined and seem that cannot be updated at runtime. The reviewed models can be classified as static, which involves that the users' privileges are predefined and remain the same across database access sessions, unless they are set manually by the database security administrator. Indeed, data in graph-oriented databases are described as dynamic because they do not require a predefined schema and can accommodate evolving data structures. This flexibility allows for different records to have varying attributes and enables easy schema changes without disrupting existing data. Enforcing access control policies in static models may pose challenges in ensuring compliance with security requirements and preventing unauthorized access or privilege escalation. Without mechanisms for real-time monitoring and adaptive enforcement, static models may struggle to address emerging security threats or policy violations effectively. To deal with this issue, we introduce in this paper a new role-based model

for dynamic access to graph databases.

3. PROPOSED APPROACH

3.1 Principle

We detail a novel approach based on graph-oriented databases. This approach is suitable for any control mechanism, ensuring that only trusted and authorized users can access and process data stored in a graph-oriented database. However, our model is designed to be dynamic contrary to the models having been proposed before, so, user's privileges are updated dynamically after revising some security aspects including the behavior of the user. Thus, the proposed model allows to define and to represent inherent security and structural aspects for graph-oriented databases. In Figure 1, we introduce an informational architecture of the proposed meta-model, via a UML class diagram. The entity "Graph DB" has a name attribute. It exclusively owns all the edges and vertices through the classes, "Vertex" and "Edge".

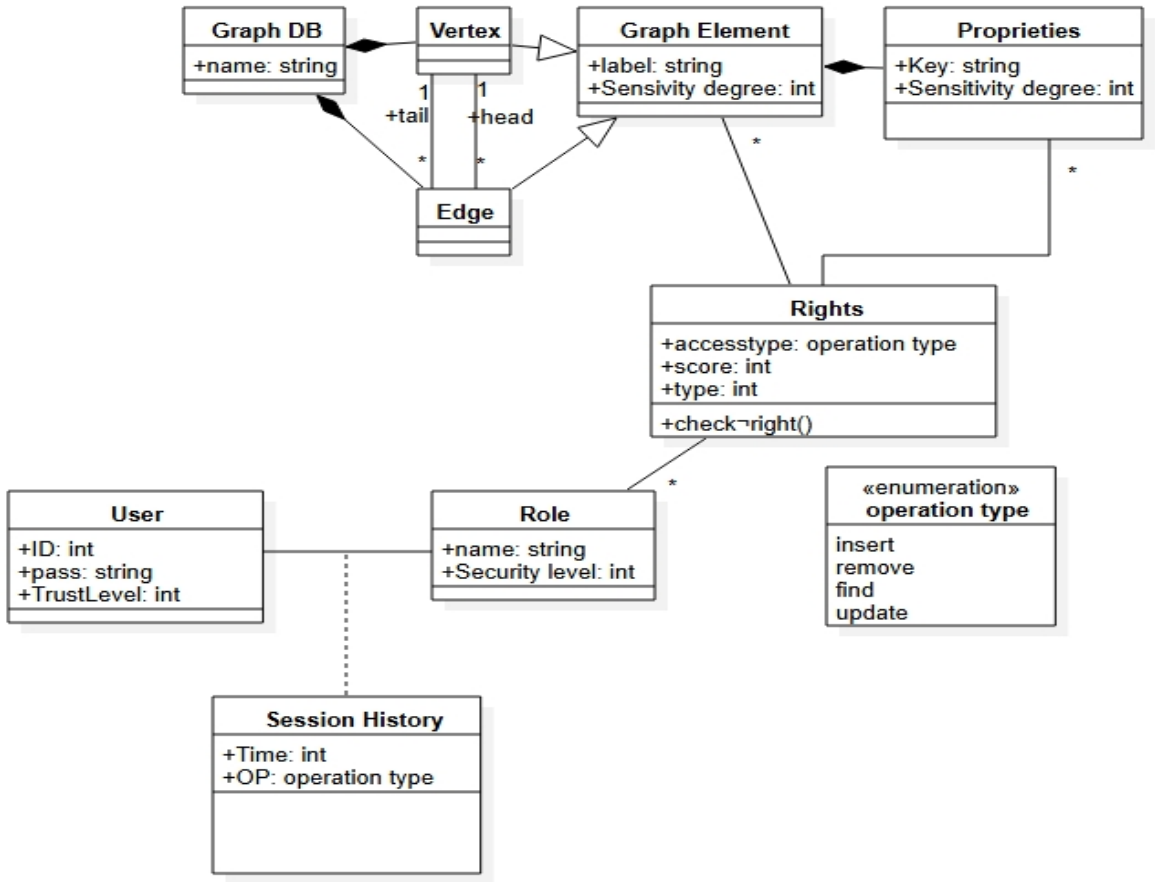

Figure 1. Trust-based access control model

Head and tail associations associate Vertex and Edge. They express the fact that an edge has one tail vertex and one head vertex. In addition, a vertex has incoming edges and outgoing edges. Obviously, "Vertex" and "Edge" are both subtypes of "Graph Element". The class "Graph Element" defines a set of labels describing the element type, in addition to a set of "Properties". The Class" Properties" has a key (property's name). The attribute "sensitivity degree" expresses the recorded data sensitivity degree. Three different degrees are defined: the lowest security degree (SD=1), a sensitive data degree (SD=2), and a very sensitive data degree (SD=3).

The proposed model is based on trust role-based access control policy. Based on this policy, users are hierarchically clustered according to their roles and regarding some considered security aspects. The classes "User" and "Role" represent, respectively, the authorized users and the roles to which they are assigned. According to their functions, users have specific rights and operations that can be performed by a given role. The attribute "TrustLevel" define the reputation degree of a given user. The security level attribute in the class role is calculated by the max value of sensitivity level of protected objects defined for a given role. The class "Rights" is an association that specifies the permission or access rights granted to a user to perform specific actions on protected objects (vertex or edge) according to a given role. The access control is a fine-grained access control data that protect vertices and edges at the attribute level. So, it will be possible to also define restrictions on properties. At the first time, a user must introduce his identifiers to register if he applies to access the database. Thus, the system affects to this user an initial reputation value. Later, the system decides if a user can be allowed to access the database after the value of his reputation has been updated. The system continuously updates the user's reputation value, after the user has accessed and used the database. Reputation value updates of a user rely on his behavior, his histories, and his operations. After each update, if the reputation credit is less than the sensitivity degree associated to the current role, the user loses the previous role to which he was assigned. Hence, he will be

required to request a role with lower privileges. However, if the trust value is above the sensitivity degree of given role, the user will be permitted to obtain the requested role, or to maintain it if it was the previous one for which he is assigned. Figure 2 shows a flow chart of how a user is checked before accessing the database.

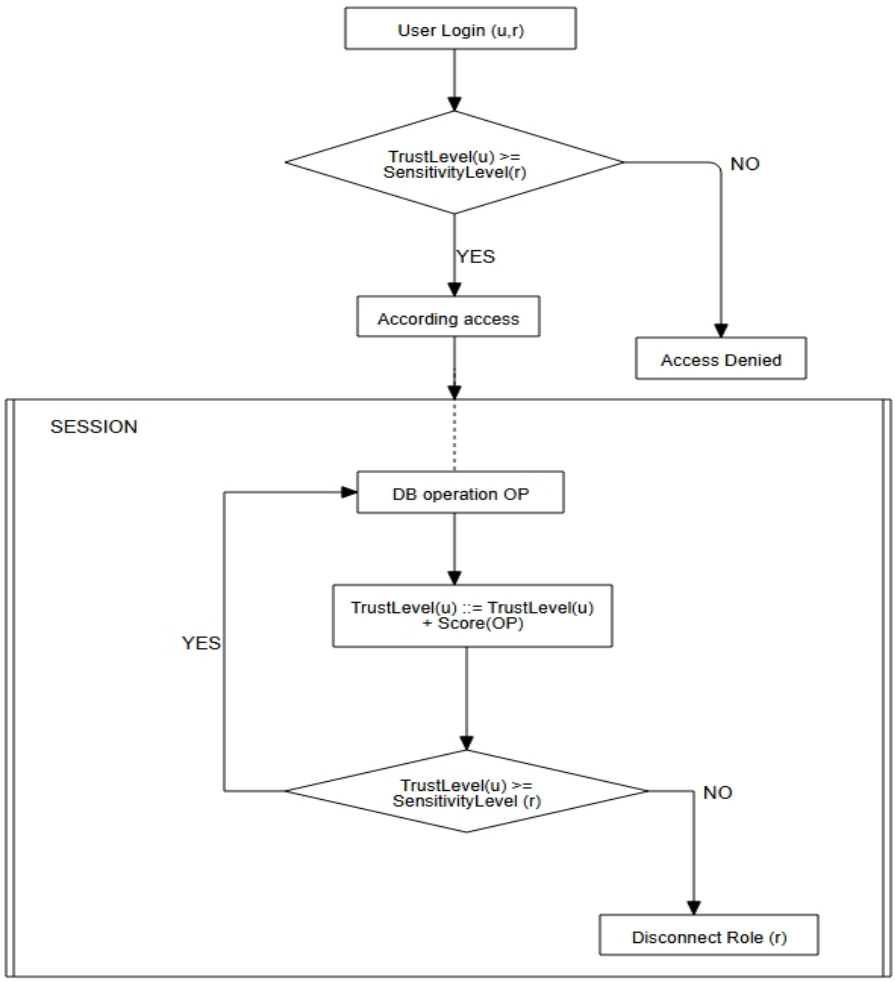

Figure 2. User access process flow chart

3.2 Trust level update

According to the graph meta-model and the flow chart introduced respectively in Figures 1 and 2. A score is affected to each performed operation that depends on the role that the user currently has, and on the type of the operation itself. So, the user's trust level under a given role can be calculated according to the Score attribute on Right class. During his connection session to the database, if the user's trust level falls under a predefined role's sensitivity level, the user is immediately denied access to the database, and therefor asked to reconnect with a role that the privileges are lower than the previous one, or he should contact the database manager to deal with the occurred incident. At the logging to the database, a requested role can be assigned to the user if only his trust level is above the sensitivity level of the requested role. According to our model the user's trust degree is computed as follows:

$$TL(u) = TL(u) + \sum_{op \in session} Score(R, op) \qquad (1)$$

During or at the end of a given session, the new trust level of a given user u is updated by adding the sum of scores corresponding to the operations that he has performed during his last session.

"*Score*(*R*,*op*)" expresses the criticality of an operation "*op*" when the user has the role *R,* that is defined according to the data integrity and protection policy. Indeed, in any database, deleting and updating items are more critical than operations that allow browsing or displaying data from a database.

3.3 Computation

The algorithm "Periodical_Check", introduced below, allows user checking for all users that are currently connected to the database. Namely, a user can be allowed to maintain his connection to the database with his current role. However, a user can be disconnected and then asked to reconnect with a role that has lower privileges, or he is invited to contact the database manager to handle the encountered security issue.

Algorithm 1: Periodical_Check

1). for any connected user u do
2). NewTL= u.TrustLevel+$\sum_{op \in session}$ Score(R,op)
3). If R.Securitylevel > NewTL then
4). SendMessage (u, " Access denied");
5). u.Disconnect
6). end if
7). end for

In the algorithm described above, all the connected users are periodically checked to verify if each one can remain connected, it should be disconnected and asked to be reconnected using a new role with low privileges (instructions 4 and 5). It can be noticed that the security level is assigned to the role and not to the user.

The algorithm "Access_Check" shows how history of operations of a user is used to assign them the role they require.

Algorithm 2: Access_Check (u: user, R: role)

```
1). if R.Securitylevel >NewTL then
2). SendMessage (u, " Access denied");
3). else
4). u.role =R;
5). u.Connect;
6). Endif
```

The instructions 1 to 2 test and decline access to the user u, given that his current trust level is below the threshold of the security level of the required role R. The instructions from 3 to 6, are executed if the current trust value of the user is above the threshold of the current role security level, then the user will be connected with the required role R.

4. EXPERIMENTATION

As far as we know, there is no work having dealt with automatic trust evaluation and used to dynamically allow role assignment of users according to their behavior. So, we aim via the current comprehensive experimental protocol to introduce a proof of concept (PoC) of the proposed model, and show that it allows to dynamically controlling access in graph-oriented database. Obviously, there are no sufficiently similar experiments that allow us to proceed to any comparative study.

4.1 Use case

To show the feasibility and the interest of the proposed model for adaptive database access control, we considered a use case related to interactions in social networks, using Noe4j NoSql database. Neo4j is an open source graph database management system developed in Java by Neo Technology Company. The product has been around since 2000, version 1.0 was released in February 2015. Neo4j allows data to be represented as nodes connected by a set of arcs, these objects having their own properties. Properties are made up of a pair of key-values of a simple type such as character strings or numeric which can be indexed. There is no need to use keys in Neo4j, because relationships have their own existence [31]. On a social network platform, a user may be a “Creator of posts” (CR), a “Commentator of posts” (CM), or “Deprived from Comments” (DP) (respectively from “Creation of posts”). In order to automatically assign roles to users of the platform, a community rating system is set up, that is based on the number of "likes", or "dislikes" that a user will receive on all the posts he created or commented on. When creating its account, a user is assigned the role of “Commentator of posts”. Then, two scenarios are possible for this user: Over time, if his "like" score is high enough, he is granted the role of "Creator of posts". However, if the "dislike" score is high enough, he is downgraded to the role of "Deprived of comments", according to the current role of user, and the feedback from readers. Table 1 describes the scoring scheme used for updating user scores:

Table 1. Score table according to the role of the user and the feedback of the readers

Feedback Role	Like	Dislike
Creator of posts	***+1***	***-1***
Commentator of posts	+0.5	-0.5
Deprived of comments	+0.5	-0.5

At the creation of his account, a user u will have a score S of 0 points. This variable S represents the trust level in our model: *TL(U,R)*:

If feedback== "like" then $S \leftarrow S+Score(R,\text{“like”})$.

If feedback== “dislike” then $S \leftarrow S+Score\ (R, \text{“dislike”})$.

The change of role according to the score of the user is done according to the following rules:

If $S \geq T$ then $R \leftarrow$ “Creator of posts”.

If $S \in [-T+1..T-1]$ then $R \leftarrow$ “Commentator of posts”.

If $S \leq -T$ then $R \leftarrow$ “Deprived of comments”.

where, T is a role conversion threshold. It will be defined experimentally using expert user profiling, and based on statistics on the number of users and the number of feedbacks to posts and comments. For our case, we have set T=Number of users / 10. This means that the difference in favor of the user among his pairs is 10%. For our use case we consider three users u_1, u_2, and u_3. Each user has an identifier and a user profile. The user profile contains, among several records, the current role of the user and his score of points, representing his level of trust on which the assignment of the role is based. The following sub-graph in Figure 3 represents a sample of the database subschema related to role assignment based on *TL* trust level for the user u_1.

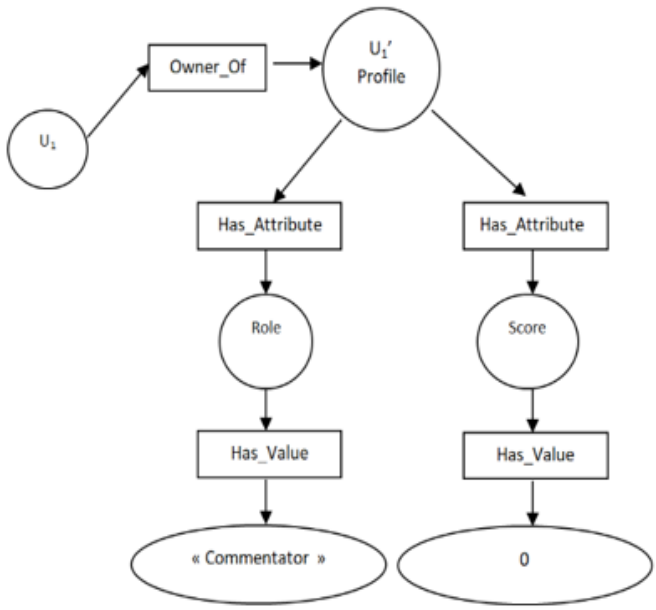

Figure 3. Sub-graph in the database corresponding to the user u_1 (resp. users u_2 and u_3)

We consider the following feedback scenarios:

- u_1 created a post P for which he had T ‘dislikes’.
- u_2 posted a comment C_1 on post P and got a total of T 'likes'.
- u_3 posted a C_2 comment on post P and got $T/2$ ‘likes’.

According to the considered scenario, the trust levels of the three users changed as mentioned in Table 2.

Table 2. Variation of the trust levels at the beginning and at the end of the considered scenario

User	Score: TL (U,R)	Role R
Trust level at the creation of users		
u_1	0	Commentator
u_2	0	Commentator
u_3	0	Commentator
Trust level at the end of the considered scenario		
u_1	-T	Deprived
u_2	+T	Creator
u_3	T/2	Commentator

4.2 Adaptive role assignment

To show the adaptive power of the proposed access control model, we have tested the scoring system by considering 50 users who interact for a period of 10 hours divided each one into 20 epochs. We reported after every 30 minutes the number of users per role. Obviously, at experimentation we assume that some users should have Creator role, so posts can be created, and then commented. Furthermore, the set of users is divided into two categories: The first one is for the users having good behavior, where they correctly post messages or comments. The second category is composed of users with bad behavior, which post or comment messages incorrectly. In our experimentation, we have considered 20% of the users (10 among 50) have bad behavior. So, at the end of the experimentation, we expect that the users with bad behavior will be deprived from interaction. Figures 4 and 5 show the overall result of the experiment, where one can notice the variations of the populations of the different roles that the users have.

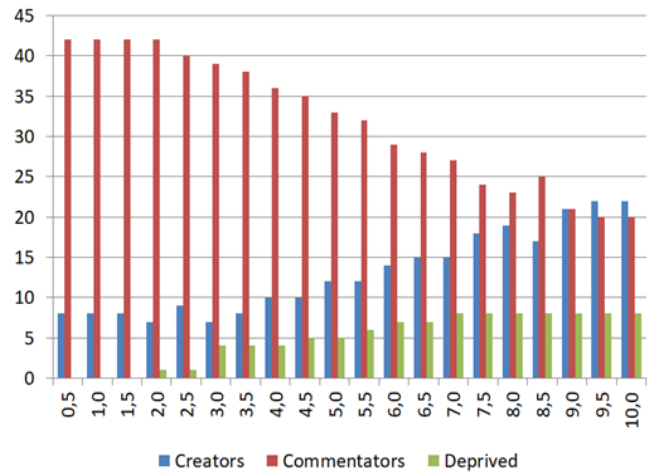

Figure 4. Numbers of roles within a set of 50 users during 10 hours of experimentation

At the beginning, some of users are "Creators", so the messages start to be posted and then commented by Commentators and also Creators. Moreover, no user was initially "Deprived", however, over time some of them become "Deprived", because they likely post or comment incorrectly.

At the end, 22 users are with the role "Creator", 20 with the role "Commentator" and 8 with the role "Deprived". Indeed, in the conducted experimentation 12 users exhibiting bad behavior. Thus, 8 of them which have initially "Creator" or "Commentator" roles are converted to "Deprived" role. Four users exhibiting bad behavior remained in the "Commentator" role because they did not interact sufficiently. No one with bad behavior has remained with "Creator" role. Such statsitic show that the bahavior-based trust computation and automatic role assignement allows to correctrly discard bad users and maintain good ones conencted to the database.

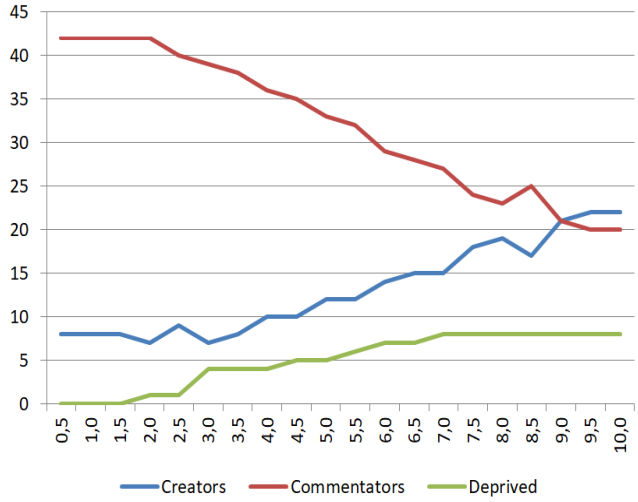

Figure 5. Variation curves of each role during experimentation time

Figure 6 shows the average numbers of users at the beginning of the session and at the end. At the beginning, no user is considered deprived, and where 16% of users were creators and 84% were commentators. At the end of the session, 26% are creators, 64% are commentators, and 10% were deprived. It can be noticed also from the various charts that the number of users per role become likely stable at the end of the session (22 creators, 20 commentators and 8 deprived), corresponding to the flat curves in Figures 4 and 5. This fact indicates that, over time, user behavior can be determined by the actions they perform within the system.

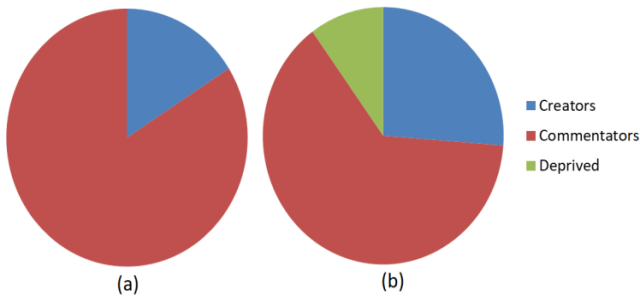

Figure 6. Average numbers of roles at the beginning (a) and at the end (b) of the exploitation period

We conclude that the experiment conducted according to the introduced use case has allowed us to validate that the proposed model and its associated computations ensure automatic and adaptive control access to the database. Without administrator intervention the model was able to automatically select the appropriate role of the users according to their behavior. Furthermore, we can notice an early convergence of the curves, which indicates that the proposed model allows an accurate but also a fast trust-based

access role assignment.

5. CONCLUSION

After Big Data has emerged in all the fields of information and communication technologies, data become more sensitive and should be so well protected. In the past, database engineers and managers focus on functionality aspects, aiming to provide first efficient databases. Despite its importance, security and privacy in databases have remained a secondary concern for several decades. To deal with security and privacy in the new advanced databases, this work has introduced a dynamic access control model in graph-oriented databases. By representing operation security and reputation of users, the model provides an effective dynamic access control, so, the graph-oriented database security is enhanced. The evaluation of the proposed model was done by conducting realistic scenarios, where the results show the effectiveness of the proposed model to automatically and adaptively control access to the graph database. The proposed model can be applied to graph databases in order to automatically control the access to the database by dynamically updating the trust level of the users. This allows the prevention of malicious behaviors, therefore enhancing database security. However, it should be noticed that the approach we have adopted requires sophisticated modeling of user behaviors, that we consider as a main limitation. In future work we expect to introduce ground truth data as learning datasets and use them with advanced machine-learning models in order to boost the performance of the proposed model, and better represent user behaviors.

REFERENCES

[1] Sahay, R. (2020). Relational databases. In Microsoft Azure Architect Technologies Study Companion. Apress, Berkeley, CA, USA, pp. 717-739. https://doi.org/10.1007/978-1-4842-6200-9_21

[2] Elmasri, R., Navathe, S. (2016). Fundamentals of Database Systems, Seventh Edition. Hoboken, NJ, USA: Pearson.

[3] Desamsetti, H. (2020). Relational database management systems in business and organization strategies. Global Disclosure of Economics and Business, 9(2): 151-162. https://doi.org/9.151-162.10.18034/gdeb.v9i2.700

[4] Sathiya Vadivoo, D.V., Shanthini, S., Vinora, A., Mohana Priya, G. (2017). An overview of database management systems and their applications along with the queries for processing the system. SSRG International Journal of Computer Science and Engineering, 4(3): 1-4. https://doi.org/10.14445/23488387/IJCSE-V4I3P101

[5] Harrington, J.L. (2010). SQL Clearly Explained, Third Edition. Morgan Kaufmann. https://doi.org/10.1016/C2009-0-61592-0

[6] Chavan, H., Shaikh, S. (2022). Introduction to DBMS: Designing and Implementing Databases from Scratch for Absolute Beginners. BPB Publications. https://doi.org/10.0000/9789355510266-001

[7] Husain, M.S., Khan, M.Z., Siddiqui, T. (2023). Big data concepts, technologies, and applications. Auerbach Publications. Auerbach Publications, New York. https://doi.org/10.1201/9781003441595

[8] Balusamy, B., Kadry, S., Gandomi, A.H. (2021). Big Data: Concepts, Technology, and Architecture. John Wiley & Sons.

[9] Harrison, G. (2015). Next Generation Databases: NoSQL, and Big Data. Apress, Berkeley, CA, USA. https://doi.org/10.1007/978-1-4842-1329-2

[10] Meier, A., Kaufmann, M. (2019). SQL&NoSQL Databases: Models, Languages, Consistency Options and Architectures for Big Data Management. Springer Vieweg Wiesbaden. https://doi.org/10.1007/978-3-658-24549-8

[11] Kshetri, N. (2014). Big data′s impact on privacy, security and consumer welfare. Telecommunications Policy, 38(11): 1134-1145. https://doi.org/10.1016/j.telpol.2014.10.002

[12] Thuraisingham, B. (2015). Big data security and privacy. In CODASPY '15: Proceedings of the 5th ACM Conference on Data and Application Security and Privacy, San Antonio, Texas, USA, pp. 279-280. https://doi.org/10.1145/2699026.2699136

[13] Thamizhiniyan, C.S., Kapuluru, C.S., Balakiruthiga, B. (2025). Mitigating NoSQL injection vulnerabilities: Techniques, examples, and best practices. In 2025 International Conference on Emerging Systems and Intelligent Computing (ESIC), Bhubaneswar, India, pp. 112-117. https://doi.org/10.1109/ESIC64052.2025.10962645

[14] Diaz, C. (2022). Database Security: Problems and Solutions. Mercury Learning & Information.

[15] Nayak, A., Poriya, A., Poojary, D. (2013). Type of NOSQL databases and its comparison with relational databases. International Journal of Applied Information Systems, 5(4): 16-19.

[16] Pokorný, J. (2011). NoSQL Databases: A step to database scalability in Web environment. International Journal of Web Information Systems, 9(1): 278-283. https://doi.org/10.1145/2095536.2095583

[17] Bhogal, J., Choksi, I. (2015). Handling big data using NoSQL. In 2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops, Gwangju, Korea (South), pp. 393-398. https://doi.org/10.1109/WAINA.2015.19

[18] Aldwairi, M., Jarrah, M., Mahasneh, N., Al-khateeb, B. (2023). Graph-based data management system for efficient information storage, retrieval and processing. Information Processing & Management, 60(2): 103165. https://doi.org/10.1016/j.ipm.2022.103165

[19] Kotiranta, P., Junkkari, M., Nummenmaa, J. (2022). Performance of graph and relational databases in complex queries. Applied Sciences, 12(13): 6490. https://doi.org/10.3390/app12136490

[20] Bertino, E., Ghinita, G., Kamra, A. (2011). Access control for databases: Concepts and systems. Foundations and Trends® in Databases, 3(1-2): 1-148. https://doi.org/10.1561/1900000014

[21] Mohamed, A.K.Y.S., Auer, D., Hofer, D., Küng, J. (2024). A systematic literature review of authorization and access control requirements and current state of the art for different database models. International Journal of Web Information Systems, 20(1): 1-23. https://doi.org/10.1108/IJWIS-04-2023-0072

[22] Morgado, C., Baioco, G.B., Basso, T., Moraes, R. (2018). A security model for access control in graph-

oriented databases. In 2018 IEEE International Conference on Software Quality, Reliability and Security (QRS), Lisbon, Portugal, pp. 135-142. https://doi.org/10.1109/QRS.2018.00027

[23] Blanco, C., García-Saiz, D., Peral, J., Maté, A., Oliver, A., Fernández-Medina, E. (2018). How the conceptual modelling improves the security on document databases. In 37th International Conference on Conceptual Modeling, Xi'an, China, pp. 497-504. https://doi.org/10.1007/978-3-030-00847-5_36

[24] Kulkarni, D. (2013). A fine-grained access control model for key-value systems. In Proceedings of the Third ACM Conference on Data and Application Security and Privacy, San Antonio, Texas, USA, pp. 161-164. https://doi.org/10.1145/2435349.2435370

[25] Clark, S., Yakovets, N., Fletcher, G., Zannone, N. (2022). ReLOG: A unified framework for relationship-based access control over graph databases. In IFIP Annual Conference on Data and Applications Security and Privacy, Newark, NJ, USA, pp. 303-315. https://doi.org/10.1007/978-3-031-10684-2_17

[26] Ahmadi, H., Small, D. (2019). Graph model implementation of attribute-based access control policies. arXiv preprint arXiv: 1909.09904. https://doi.org/10.48550/arXiv.1909.09904

[27] Magomedov, S., Pavelyev, S., Ivanova, I., Dobrotvorsky, A., Khrestina, M., Yusubaliev, T. (2018). Anomaly detection with machine learning and graph databases in fraud management. International Journal of Advanced Computer Science and Applications, 9(11). https://doi.org/10.14569/IJACSA.2018.091104

[28] Alotaibi, A., Alotaibi, R., Hamza, N. (2019). Access control models in NoSQL databases: An overview. Journal of King Abdulaziz University (JKAU), 8(1): 1-9. https://doi.org/10.4197/Comp. 8-1.1

[29] Paneque, M., del Mar Roldán-García, M., Blanco, C., Maté, A., Rosado, D.G., Trujillo, J. (2024). An ontology-based secure design framework for graph-based databases. Computer Standards & Interfaces, 88: 103801. https://doi.org/10.1016/j.csi.2023.103801

[30] Chabin, J., Ciferri, C.D., Halfeld-Ferrari, M., Hara, C.S., Penteado, R.R. (2021). Role-based access control on graph databases. In 47th International Conference on Current Trends in Theory and Practice of Computer Science, Bolzano-Bozen, Italy, pp. 519-534. https://doi.org/10.1007/978-3-030-67731-2_38

[31] Kemper, C. (2015). Building an Application with Neo4j. In: Beginning Neo4j. Apress, Berkeley, CA, USA, pp. 103-129. https://doi.org/10.1007/978-1-4842-1227-1_8