

Towards an Access Control Model for Knowledge Graphs

(Discussion Paper)

Marco Valzelli¹, Andrea Maurino¹, Matteo Palmonari¹ and Blerina Spahiu¹

¹*Department of computer systems and communication,
University of Milano - Bicocca, Milano, Italy*

Abstract

Nowadays Knowledge Graphs are a common way to integrate and manage large amount of information. This involves also sensitive domains like security, so the management of access control on these graphs became crucial. Due to their dimension, Knowledge graphs are often stored using NoSQL solutions, that have very poor support to access control. In this paper a distributed and secure Knowledge graph management system is presented. The system supports both open and closed access control and its architecture guarantees the secure access of very large knowledge graph by means of query transformation.

Keywords

access control, knowledge graph, NoSQL, security

1. INTRODUCTION

In 2012 Google introduced the Knowledge Graph (KG in the follow) as a new technology to improve its famous search engine. Basically, it is a graph that contains all the main named-entity of the world and the relations between them, so that the search engine is able to identify the subject of the query and suggest all object or features related to it. Since their introduction, KGs have been largely used by a lot of big and small companies (e.g. Amazon, Facebook and so on), but they are also used in several research domains as a way to describe in a common and shared way the knowledge of a given field. Several companies are now aggregating the knowledge stored in both structured and unstructured sources into a unique, comprehensive KG, for a better data management. It is worth noting that a KG can be considered as an evolution of the traditional data integration approach, and as a consequence the size of KG is bigger than previous solutions.


As the uses case of KG increase, they start to include also very sensitive data like for cybersecurity purposes [1], in the military sector [2] the law enforcement sector [3], but also civilian cases like the medical one [4, 5]. For example in this last case, the KG includes all information related to doctors, patients, diseases and so on. Some people may access the KG in order to analyze correlation among clients but they cannot access to personal details of patients, while

SEBD 2021: The 29th Italian Symposium on Advanced Database Systems, September 5-9, 2021, Pizzo Calabro (VV), Italy

✉ m.valzelli@campus.unimib.it (M. Valzelli); andrea.maurino@unimib.it (A. Maurino); matteo.palmonari@unimib.it (M. Palmonari); blerina.spahiu@unimib.it (B. Spahiu)



© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

another person e.g. a doctor can access his patients' information but s/he cannot see information about others doctors. So, the explosion of KG raises new issues wrt. the preservation of relevant information from unauthorized access and their growing request for scalability at a higher level.

Currently, there is no common opinion about how to represent a KG. Among others, RDF is one of the most adopted description languages, SPARQL is the query language for RDF model and many systems for managing RDF data (also called triple store) are available. However there are proposals to implement an access control mechanism in RDF [6, 7, 8], but they are not scalable nor flexible and, most important not implemented in commercial triple store. Recently, others graph models like the Labelled Property Graph model (LPG from now on) has been used to represent knowledge graphs.

In this extended abstract, we propose a solution for enabling access control of large KG in a scalable way. The scalability issue is solved by means of Janusgraph¹ a distributed and scalable property graph. We provide a systematic translation of RDF concepts in terms of property graph model. Then we define an access control method based on logical description of KG and users in terms of property graph model; our solution support a subset of SPARQL query languages by optimizing Gremlinator [9] a software layer able to translate a SPARQL query into a property graph query.

The paper is organized as follows: in section 2 we report the most important contribution of literature and industrial solution for the access control problem applied to graphs, while in section 3 we provide a more formal description of the problem. Section 4 presents the proposed access control method and in section 5 the implementation of the model is shown. Finally in section 6 we draw conclusion and discuss future works.

2. Related Work

The great success of graphs as data model is very recent, but in the past we can find that trees, a particular type of graph, have been used as a data structure on which apply access control. With the need to store data comes along the need to provide an access control framework. An example is XML, a format widely used to exchange information across the web, on which has been proposed an access control model[10]. Starting from 2001 with the birth of the semantic web [11], the RDF semantic graph framework provided an efficient way to store big amounts of data in a graph format. RDF datasets usually are meant to be shared across the web, so no access control models have been provided. Since the users of RDF endpoints are often anonymous, the literature offers many proposals of context-based access control frameworks. One of these [12] also suggests an interesting mechanism for the creation of view on the graph based on the composition of allow/deny rules. The researches in this area also proposes to define policies through ontologies and to apply these policies using ontological reasoning. Research in this area also tried to address particular aspects of this technology like distribution [6, 13], federation [14] inference of new information [15] and possible solutions to the scalability problem [7]. Also more traditional access control models like DAC (Discretionary Access Control) have been implemented [8]. Although there are significant literature for access control in RDF Graphs [16], these approaches cannot be directly applied in other graph contexts as they focus on different

¹janusgraph.org

aspects or heavily rely on ontological reasoning, that is usually provided only by native RDF stores.

Available NoSQL tools were not designed with security and privacy criterion and still lack in this aspect [17, 18]. Just a small part of these solutions support access control features, and only for specific types of policies, like RBAC (Role-Based Access Control). Also notable is the case of Accumulo² where access control at cells level is provided.

Graph model is supported by several NoSQL products, but in a very different way from RDF systems. While the letters are based on triples, the former adopt the property graph model. Graph database can be divided into native, where data is physically stored in terms of nodes and edges, and the hybrid, that provides a property graph model but rely on a different storage model like the wide-column.

Neo4J is the most common native graph database and it implements the LPG model. It still has only basic security and access control features: it allows to define user role and their privileges³, but not at a fine-grained level.

The only significant work in the property graph context [19] has designed an access control model defining how users or groups of users can access and manipulate data. However, it has some drawbacks like lack of flexibility and redundancy. In fact, you can only describe positive permissions and you have to define them for every node, so it's not clear how the model can be applied with inheritance and conflicts. Furthermore, this work and the neo4j proposal has both the characteristic that they requires quite complex algorithm to be evaluated, so their scalability is in doubt. A different approach has been proposed in the Tinkerpop community⁴ where the evaluation mechanism of authorizations has been integrated in the Graph-DMBS but it checks for access control metadata in every edge and node, so it needs a certain redundancy both in the stored data and in the interrogation, impacting on the performance of the system.

3. Considered Use Cases

In the past years the literature on access control, thanks to a great adoption in commercial DBMS, has defined a variety of policies one can implement[20]. We focus on three main kind of policies for access control, that are: **DAC**, **MAC**, and **RBAC**. The former provides for a direct rights specification between users and resources. The letter allows to assign rights to users based on their roles, while MAC provides access to the resources based on the clearance level of the user, following the need-to-know principle (closed policy): a subject should only be given access rights that are required to carry out the subject's duties.

These access control policies were designed to be mutually exclusive, but instead we want to combine and use them at the same time. We extend it by adding more abstraction also over resources, borrowing the notion of classes from RDF ontologies. In fact, groups of users and roles are ways to assign the same rights to more users at the same time, saving memory and maintainability time. We pursue the same objective over resources by giving the possibility to define access rights over entity classes and then propagate them using inheritance mechanisms.

²<https://accumulo.apache.org/>

³neo4j.com/blog/role-based-access-control-neo4j-enterprise/

⁴archive.fosdem.org/2019/schedule/event/graph_access_control_tinkerpop

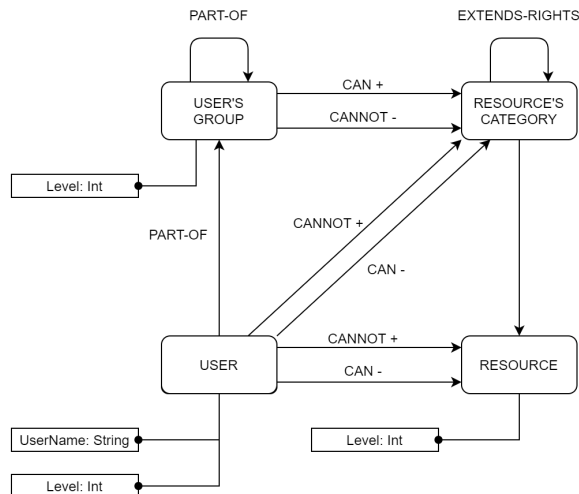


Figure 1: Graph pattern for the access control framework. "Can" and "Cannot" edges cannot be present simultaneously

For example, let's consider the case of a national security agency fighting terrorism. It KG includes classes like Person, Organization and Work. We call this extended graph of the domain Graph. It should be noted that we can use relations between classes as a way to extend access rights between instances that belongs to those classes. In fact, if is defined a relation between "Criminal" and "Document" classes and a user has access to a specific criminal profile, probably she will have access also to his related documents (except for clearance). It is possible to represent as a graph also users, that can be grouped by departments of the agency or their roles. We call this part of the graph the User's graph. According to our scenario, we want to specify only what a user can view, adopting a closed policy. That can be done on the base of both her role and groups, as foreseen by RBAC policy. Probably, different departments of the organization investigate different areas of the world, so they have to be able to view only a small portion of the graph. In addition, we may want to cover also particular cases (like temporary operation) that allows access to specific documents, by means of DAC rules. Moreover, all those rules have to match a user's clearance level (MAC).

4. Access Control Model

The proposed access control model has the goal to design a security mechanism that is reusable for many different policies, especially both the cases of open and closed policies. Our approach is focused on giving or not access to resource. More specific rights like own, delete, modify can be built on the top of our model.

The idea behind our model is to exploit all the flexibility of the LPG model so we do not give a rigid data structure, but only some graph patterns to follow, as can be seen in figure 1. Three are the main part of our model: the user's subgraph, the resource's subgraph and the authorization's edges between them.

In the user's subgraph users are represented by nodes with properties such as the clearance level. They can be grouped in groups by linking them to users' group nodes. This mean they will inherit the access rights of the group. Then, groups can be grouped themselves iteratively. In the same way users can be grouped by roles' node. Potentially these types of nodes can coexist, but this will lead to a more complex rights administration.

As we mentioned before, we use relation between classes as a way to extends rights between instances. In fact, it is a common situation that some objects are semantically dependents to another object, that we call resource category. Our intuition is to use this kind of relation between classes to automatically insert specific "extends_rights" edges between their instances.

Finally, we can specify access rights between users or user's groups and resource's classes with authorization edges. In our model we want to allow both open and closed policies, so these edges can have opposite meanings. With closed policies access' rights are usually assigned with the "need to know" principle, which is that a user has to be given access to the resources strictly necessary to carry out his duties, and everything else is forbidden. On the contrary, using open policies are specified resources where a user has not access and all that is not specified is allowed.

In addition, our model provides for particular policy specification using exception edges, that are authorization edges between user and resource in contrast to those between users' groups and resources' categories. Imposing this constrain we ensure to allow expressiveness without leading to too much complexity.

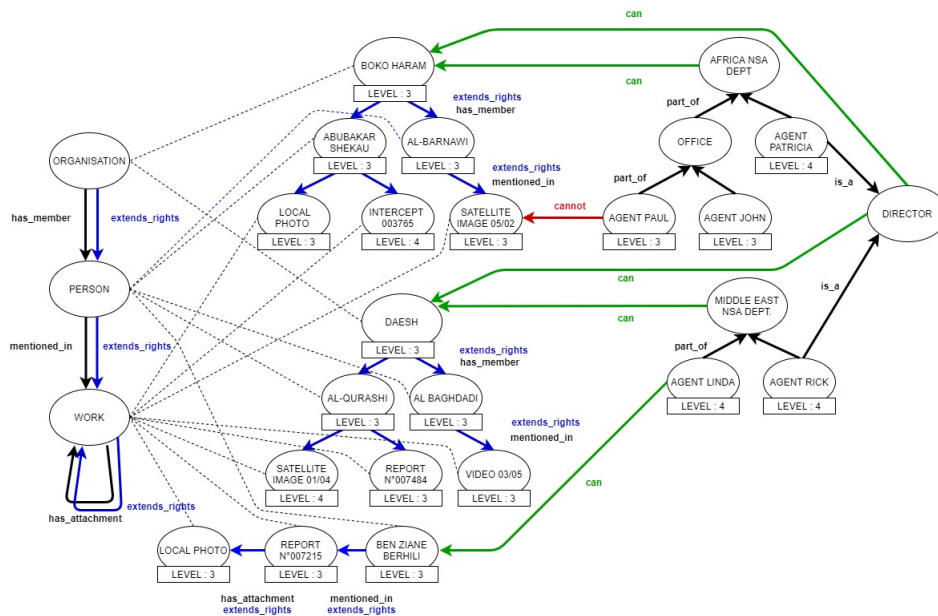


Figure 2: example of KG integrated with the users' Graph and the authorization edges

Determining which node a user can access with a traversal has many advantage:

- The graph model has great readability and also inheritance mechanism are easy to understand

- You can specify a great variety of policies with a few edges and attributes
- Full-scans are not needed as you can just use edges to reach resources you are or not allowed to access
- Maintainability is also simplified. For example, if a user wants to know which users have access to an object, she can exploit the reverse path used during the resource's access mentioned before

As an example of the above mentioned concepts let consider figure 2 whose Domain Graph is taken from DBpedia. The user's graph is linked to the Domain Graph with specific edges, and it is also integrated by AC edges itself. Let's clarify their meanings:

- Green edges represent the positive "can_view" permission.
- Red edges represent the negative "cannot_view" permission.
- Blue edges represent the extend rights relation.

This last relation can be defined between specific classes of the Domain Graph Ontology. For example, we specify that the relation "mentioned_in" extends the access rights from entity of type "Person" to entity of type "Work". Then we can use automatic procedure to add the access control attribute "extends_rights" to all the edges "mentioned_in" between entities of the classes defined before. The graph shown in figure 2 allow to describe the follow access rules:

- The department "Africa NSA" is allowed to view all information about Boko Haram organisation, but:
 - The object "Intercept 003765" cannot be seen from agent Paul and agent Patricia due to too low clearance level
 - The object "Satellite image 05/02" cannot be seen from agent Paul due to a "cannot" edge.
- The department "Middle East NSA" is allowed to view all Daesh organisation and Boko Hara is and affiliated organization.
- Agent Rick and agent Patricia are Director of their department, and this allows them to have full access to both organisation

Moreover, let assume that the agency found a connection between Daesh and the drug dealer Ben Ziane Berhily, who supplies daesh amphetamine. Agent Linda is now allowed to view Ben Ziane data with an exception specified with the "can" edge.

5. Implementation

For the implementation of our access control model, we rely on Tinkerpop⁵, which is an Apache project that provides several services and is compatible with fairly all the most common graph products. The most useful feature of this framework for our purposes is the SubGraphStrategy, which is a traversal strategy that allows to create a virtual subgraph based on given constraints. What this method really does is to provide an automatic mechanism that, given a user query,

⁵tinkerpop.apache.org/

verify at every step if the initial constraints are satisfied. Since we use a pattern-based approach integrated with exceptions, we cannot use constraints based on attributes.

Our solution is to split the process into a two step. First, we retrieve all the IDs of the resource a user can access, then we use this list as a constraint for the traversal strategy. This prevents us to check for many different conditions at every step using a more immediate check like equality between IDs.

6. Conclusions and Future Work

We find out that in the state of the art there are not efficient, flexible and general-purpose access control models for Knowledge graphs. We propose an approach based on graph traversal over specific patterns and on the creation of a subgraph using a Tinkerpop feature to address this issue. This is a preliminary model on the top of which it has to be build a more complete access control policy that includes write, delete and own rights. We left as a future work also an extensive test of the scalability of the model.

References

- [1] A. Piplai, et al., Creating cybersecurity knowledge graphs from malware after action reports (2019).
- [2] F. Liao, L. Ma, D. Yang, Research on construction method of knowledge graph of us military equipment based on bilstm model, in: International Conference on High Performance Big Data and Intelligent Systems, IEEE, 2019, pp. 146–150.
- [3] P. Szekely, et al., Building and using a knowledge graph to combat human trafficking, in: International Semantic Web Conference, Springer, 2015, pp. 205–221.
- [4] M. Rotmensch, Y. Halpern, A. Tlimat, S. Horng, D. Sontag, Learning a health knowledge graph from electronic medical records, Scientific reports 7 (2017) 1–11.
- [5] L. Shi, et al., Semantic health knowledge graph: Semantic integration of heterogeneous medical knowledge and services, BioMed research international 2017 (2017).
- [6] L. Kagal, T. Finin, A. Joshi, A policy based approach to security for the semantic web, in: International semantic web conference, Springer, 2003, pp. 402–418.
- [7] F. Abel, other, Enabling advanced and context-dependent access control in rdf stores, in: The Semantic Web, Springer, 2007, pp. 1–14.
- [8] S. Kirrane, Linked data with access control, in: Workshop on. pp, volume 14, 2015, p. 23.
- [9] H. Thakkar, D. Punjani, J. Lehmann, S. Auer, Two for one: Querying property graph databases using sparql via gremlinator, in: International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA), 2018, pp. 1–5.
- [10] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, P. Samarati, A fine-grained access control system for xml documents, ACM Transactions on Information and System Security (TISSEC) 5 (2002) 169–202.
- [11] T. Berners-Lee, J. Hendler, O. Lassila, et al., The semantic web, Scientific american 284 (2001) 28–37.

- [12] R. Stojanov, S. Gramatikov, I. Mishkovski, D. Trajanov, Linked data authorization platform, *IEEE Access* 6 (2017) 1189–1213.
- [13] J. Hollenbach, J. Presbrey, T. Berners-Lee, Using rdf metadata to enable access control on the social semantic web, in: *Workshop on Collaborative Construction, Management and Linking of Structured Knowledge*, volume 514, 2009, p. 167.
- [14] M. Goncalves, M.-E. Vidal, K. M. Endris, Pure: A privacy aware rule-based framework over knowledge graphs, in: *International Conference on Database and Expert Systems Applications*, Springer, 2019, pp. 205–214.
- [15] A. Jain, C. Farkas, Secure resource description framework: an access control model, in: *Symposium on Access control models and technologies*, ACM, 2006, pp. 121–129.
- [16] S. Kirrane, A. Mileo, S. Decker, Access control and the resource description framework: A survey, *Semantic Web* 8 (2017) 311–352.
- [17] E. Sahafizadeh, M. A. Nematbakhsh, A survey on security issues in big data and nosql, *Advances in Computer Science: an International Journal* 4 (2015) 68–72.
- [18] A. A. Alotaibi, R. M. Alotaibi, N. Hamza, Access control models in nosql databases: An overview (2019).
- [19] C. Morgado, G. B. Baioco, T. Basso, R. Moraes, A security model for access control in graph-oriented databases, in: *International Conference on Software Quality, Reliability and Security*, IEEE, 2018, pp. 135–142.
- [20] R. S. Sandhu, P. Samarati, Access control: principle and practice, *IEEE communications magazine* 32 (1994) 40–48.