A systematic literature review of authorization and access control requirements and current state of the art for different database models

Aya Khaled Youssef Sayed Mohamed, Dagmar Auer, Daniel Hofer and Josef Küng

Institute for Application-Oriented Knowledge Processing (FAW), Johannes Kepler University Linz, Linz, Austria and LIT Secure and Correct Systems Lab, Linz Institute of Technology, Johannes Kepler University Linz, Linz, Austria

Received 30 April 2023
Revised 18 July 2023
Accepted 18 August 2023

Abstract

Purpose – Data protection requirements heavily increased due to the rising awareness of data security, legal requirements and technological developments. Today, NoSQL databases are increasingly used in security-critical domains. Current survey works on databases and data security only consider authorization and access control in a very general way and do not regard most of today's sophisticated requirements. Accordingly, the purpose of this paper is to discuss authorization and access control for relational and NoSQL database models in detail with respect to requirements and current state of the art.

Design/methodology/approach – This paper follows a systematic literature review approach to study authorization and access control for different database models. Starting with a research on survey works on authorization and access control in databases, the study continues with the identification and definition of advanced authorization and access control requirements, which are generally applicable to any database model. This paper then discusses and compares current database models based on these requirements.

Findings – As no survey works consider requirements for authorization and access control in different database models so far, the authors define their requirements. Furthermore, the authors discuss the current state of the art for the relational, key-value, column-oriented, document-based and graph database models in comparison to the defined requirements.

Originality/value – This paper focuses on authorization and access control for various database models, not concrete products. This paper identifies today's sophisticated – yet general – requirements from the literature and compares them with research results and access control features of current products for the relational and NoSQL database models.

Keywords Authorization, Access control, Requirements, Relational database, NoSQL, Key-value database, Column-oriented database, Document-based database, Graph database

Paper type Research paper

This paper forms part of a special section "Recent advances in big data and security engineering for web/distributed applications part III", guest edited by Tran Khanh Dang.

The research reported in this paper was partly supported by the LIT Secure and Correct Systems Lab funded by the State of Upper Austria. The work was also funded within the FFG BRIDGE project KnoP-2D (Grant No. 871299).

International Journal of Web Information Systems
Vol. 20 No. 1, 2024
pp. 1-23
Emerald Publishing Limited
1744-0084
DOI 10.1108/IJWIS-04-2023-0072

1. Introduction

The increasing awareness of data security, legal requirements and technological developments is leading to sophisticated data protection requirements (Bertino *et al.*, 2011). Today, not only relational, but increasingly NoSQL databases are used in business and security-critical domains, due to their ability to deal with big interconnected data (Zugaj and Beichler, 2019). Security, especially authorization and access control, has not yet been sufficiently addressed.

Authorization and access control are recognized as the most important security issues in big data (Tankard, 2012). Authorization is the specification of access rights in terms of who (subject) can perform what action on which resource, while access control enforces these access rights. Access control is crucial to regulate and check the flow of information in enterprise systems. It prevents access to data by unauthorized users.

There is no general solution that applies to all database models. Each database model has different access control requirements, depending on the kind of data (i.e. structured, semi-structured or unstructured), regardless of the underlying datastore. Besides, the fine-grained access control (FGAC) solutions developed for relational database systems cannot be reused in non-relational datastores due to the schemaless nature of many NoSQL models.

The objective of this work is to guide researchers and practitioners by providing authorization and access control requirements. In addition, we study the features and limitations of the database models. In this literature work, we aim to answer the following research questions:

RQ1. What are the main findings in survey works on authorization and access control for different database models?

RQ2. What are the general requirements to apply fine-grained dynamic authorization and access control in databases?

RQ3. Are these requirements satisfied for the respective database models?

The remainder of this paper is organized as follows. In Sections 2 and 3, we provide an overview of our research method and related survey works, respectively. In Section 4, we identify general authorization and access control requirements for different database models. We discuss these requirements in terms of the relational and NoSQL database models (i.e. key-value, column, document and graph) in Sections 5–9. For each of these models, we give an overview followed by a discussion of the authorization and access control requirements ending with the features that are either supported in databases or published in research works. We then provide an overall discussion in Section 10. The paper concludes with a summary in Section 11.

2. Research method

The overall goal of our research is to understand the similarities and differences of current database models with respect to authorization and access control. As our focus is on the database models and not on specific database systems, we follow a *systematic literature review (SLR)* research method.

We start with a SLR of survey papers on authorization and access control for relational and NoSQL database models to answer research question *RQ1*. We then identify authorization and access control requirements (*RQ2*) by systematically analyzing research works, which address or contribute to enhancing authorization and access control for a specific database model. To answer *RQ3*, we study these requirements with respect to relational and NoSQL database models in detail, relying on papers and other resources from research and practice.

2.1 Systematic literature review for RQ1

- **Research question.** What are the main findings in survey works on authorization and access control for different database models?
- **Key concepts.** (1) Survey or review, (2) access control, (3) NoSQL model and (4) relational model. We excluded the terms Internet of Things (IoT) and blockchain, which came up frequently in the test search, but are not relevant in our context. The test search results show that the terms authorization and access control are not consistently used. They are often used as synonyms, but access control is usually considered to be the more comprehensive term. In addition, we refined the key concept database model to NoSQL and relational model, as the test search results with the term *database* did not discuss relational and NoSQL models within the same paper. Furthermore, we applied different filters to the search results, e.g. time span (2010–2023), science categories in Web of Science such as computer science information systems and software engineering, or reviews in Google Scholar.
- **Literature sources.** We searched the following sources: (1) Google Scholar, (2) Web of Science, (3) IEEE Xplore, (4) ACM digital library and (5) SpringerLink, combining the key concepts stated before. We scanned the sources, their references and other relevant references suggested by the tools. We then stored them in a Citavi [1] project, where we manage our references.
- **Analysis.** We analyzed the selected papers in detail, categorized them and marked the relevant parts. We identified additional references during the detailed study and added them.
- **Results.** The results of this SLR are documented in Section 3.

The studied resources have a broader focus on security in database-related aspects and only provide an overview of authorization and access control.

2.2 SLR for RQ2

- **Research question.** What are the general requirements to apply fine-grained dynamic authorization and access control in databases?
- **Key concepts.** (1) Requirement, (2) authorization, (3) access control and (4) database. We merged the concepts *database model* and *database* as the distinction was nearly irrelevant for the matches. We did not include *fine-grained* as it restricted the search space too much when applied in searches considering titles and was nearly irrelevant in full text searches.
- **Literature sources.** We used the same sources as with the SLR for *RQ1*, but started with the Web of Science. We also followed the references in the papers and recommendations by the tools as the identified keywords were not sufficient. All relevant papers are stored in Citavi.
- **Analysis.** We categorized the papers according to the requirements and selected the relevant ones in the context of data protection and data model. We ignored requirements related to protecting the database schema, policies or encryption.
- **Results.** In Section 4, we describe the identified requirements, which we consider in our comparison of authorization and access control for different data models.

To study these requirements in the context of different database models, research papers along with resources for database systems have been considered.

2.3 SLR for RQ3

- **Research question.** Are these requirements satisfied for the respective database models?
- **Key concepts.** (1) requirement and/or details to the previously identified ones, i.e. (i) granularity and fine-grained, (ii) content-based, (iii) context-based or context-aware, (iv) custom and (v) separation of concerns, external or policy-based, (2) authorization and/or access control, (3) relational, or (i) SQL, (ii) Oracle, (iii) MySQL or MariaDB or (iv) PostgreSQL, (4) NoSQL and key-value, or (i) Redis or (ii) Accumulo, (5) NoSQL and column-oriented, or (i) Cassandra or (ii) HBase, (6) NoSQL and document-based, or (i) MongoDB or (ii) Couchbase, (7) graph database, or (i) Neo4j, (ii) Microsoft Azure CosmosDB or (iii) ArangoDB
- **Literature sources.** (1) Our Citavi project, as it already contains many relevant sources, (2) Google Scholar, (3) Web of Science, (4) IEEE Xplore, (4) ACM digital library, (5) SpringerLink and (6) Google Search Engine (especially for product-specific information). We researched each model separately with respect to the different requirements, authorization and access control and only later got into the details of concrete databases (e.g. Redis). These sources are managed in Citavi.
- **Analysis.** We analyzed the sources according to the requirement and data model. In addition, we differentiated between research papers and database products and thus, between latest features in products and research results.
- **Results.** The results are organized according to the data models. We give an overview of the core characteristics of each model and provide a discussion of authorization and access control requirements based on research results and/or database products (see Sections 5–9). The findings are synthesized and compared in Section 10.

3. Related work

Several survey works address security in databases, especially NoSQL databases. For example, Sicari *et al.* (2022) is one of the very recent literature researches discussing security and privacy in the context of NoSQL database models selecting one for each category, i.e. Redis, Cassandra, MongoDB and Neo4j for key-value, column, document and graph, respectively. They compared their security features and considerations with respect to encryption, authentication, authorization and auditing. They also provided a comparison of relational and NoSQL database systems in terms of schema, redundancy, atomicity, consistency, isolation, durability, scalability and query language. This work also highlighted the need for FGAC and access control models based on the semantic content of data (refer to *R1* and *R2*).

Another recent survey work by Samaraweera and Chang (2021) discussed not only the relational, NoSQL and NewSQL database models, but also various database security risks and types of attacks. Moreover, they analyzed different data protection mechanisms and extensively compared 32 datastores from all database models in terms of authorization and access control, authentication, auditing and logging, encryption and consistency model for data integrity. Alotaibi *et al.* (2019) reviewed access control models in different NoSQL databases, highlighting the lack of FGAC. Colombo and Ferrari (2019) focused on access

control, presenting a literature review for existing access control solutions in NoSQL datastores. They also defined FGAC, context management and efficiency of access control as key requirements behind the definition of an access control mechanism for big data platforms. They classified the state-of-the-art approaches into platform-specific, platform-independent and domain-specific big data (i.e. data stream analytics and IoT). The proposed frameworks were analyzed in terms of the target platform, access control model, maximum granularity, context support and efficiency. They eventually discussed some open research issues related to big data access control.

Dindoliwala and Morena (2017) surveyed several NoSQL databases (i.e. MongoDB, Cassandra, GemStone, db4o and Objectivity/DB) and compared their authentication, authorization, auditing and data encryption features. The work presented in Zahid *et al.* (2014) performed an assessment to evaluate the security of sharded NoSQL stores in Cassandra, MongoDB, CouchDB, Redis and HBase. The assessment criteria were authentication, access control, secure configurations, data encryption and auditing. The security features of the same databases were analyzed again in Dadapeer and Adarsh (2016), in addition to defining the main security issues in NoSQL database. Sahafizadeh and Nematbakhsh (2015) included more NoSQL stores (i.e. HyperTable, Voldemort, DynamoDB and Neo4j), besides the five database systems used in Zahid *et al.* (2014), along with Dadapeer and Adarsh (2016).

However, these works had a broader focus on security features with little consideration for authorization and access control. Furthermore, they selected particular datastores to compare and analyze their features. We focus on the database model, rather than a specific database management system (DBMS).

4. Authorization and access control requirements

Data security solutions have to meet three core requirements: confidentiality (also referred to as secrecy), integrity and availability. Confidentiality refers to protection against unauthorized access, integrity ensures unauthorized and improper modification of the data, whereas availability focuses on avoiding unavailability of software and hardware, as well as recovery from errors (Bertino *et al.*, 2007; Ferrari, 2009).

In Section 3, we already discussed surveys on data protection support in various databases and models. However, authorization and access control were only marginally considered in these studies. Since authorization and access control are among the most important means of supporting confidentiality and integrity, we consider them for our detailed requirements, with focus on protecting the data in the datastores.

We define the following requirements for authorization and access control for datastores (independent of their data model):

- *R1 Varying granularity* of protected data objects.
- *R2 Content-based* authorizations.
- *R3 Context-based* authorizations.
- *R4 Custom authorizations*.
- *R5 Separation of concerns*.

4.1 Varying granularity (R1)

For more than 20 years, the demand for protecting data at different granularity levels, i.e. from sets of data objects to parts of a single one, has been heavily increasing (Bacon *et al.*, 2003; Ferrari, 2009; Bertino *et al.*, 2011; Kulkarni, 2013; Kirrane, 2015). Access control in

databases differs fundamentally from that in file systems due to the varying levels of granularity of the protected data objects in the specification and enforcement of authorizations (Bertino *et al.*, 2011). According to the data model, different entities (e.g. tables or records in relational databases, vertices and edges in graph databases, and documents in document stores) are considered in the authorization specification. The granularity of these entities can be at the level of fields, attributes or data components, based on the model structure. Fine-grained access control is a fundamental requirement today (Bertino *et al.*, 2007; Bertino and Sandhu, 2005; Colombo and Ferrari, 2019). However, new challenges emerge due to heterogeneous data in schema-free NoSQL datastores, where authorizations and their enforcement cannot rely on a known schema like in relational databases (Okman *et al.*, 2011; Sahafizadeh and Nematbakhsh, 2015; Colombo and Ferrari, 2017, 2019; Sicari *et al.*, 2022). Thus, defining and enforcing fine-grained authorization policies in NoSQL datastores is still challenging.

4.2 Content-based authorizations (R2)

Content-based access control is considered an important requirement to protect data in a datastore (Bertino and Sandhu, 2005). Access decisions rely on data content, i.e. metadata (Bertino and Sandhu, 2005) or any content from the datastore (Bertino and Sandhu, 2005; Bertino *et al.*, 2007). Therefore, not only the structure, but also the semantics of the data are considered in authorizations and their enforcement (Bertino *et al.*, 2011; Kulkarni, 2013; Sahafizadeh and Nematbakhsh, 2015; Sicari *et al.*, 2022).

Consider a document-based database, where a document is the unit of storing data. The document consists of fields with keys and values, which can be used to specify authorizations, e.g. a rule in an authorization policy. When the rule defines an access condition on a specific field, only data satisfying this rule can be accessed. This requirement not only covers the use of content-based restrictions on resources (objects), but also on attributes of requesters (subjects), such as their job title, specific qualification or signing authority (Bertino *et al.*, 2007; Kirrane, 2015).

4.3 Context-based authorizations (R3)

Context information is associated with the user (subject) requesting access and/or the resources to be accessed. The authorization policy is evaluated to decide about access. Thus, dynamic user privileges are addressed in authorization policies to allow constraints on, e.g. geographical locations, time or history (Bacon *et al.*, 2003; Bertino and Sandhu, 2005; Bertino *et al.*, 2007; Kulkarni, 2013; Colombo and Ferrari, 2019; Sicari *et al.*, 2022). For instance, some resource can be only accessed if the user is in the office, within the office hours, or if the user already performed some specific task in the past. Bacon *et al.* (2003) also considered more specific contexts, such as the name of the computer or a certain relationship (e.g. patient_under_care) between the user (e.g. a doctor) and the resource (e.g. patients and their medical history).

Although the database model has no impact on the user context, not all database systems, especially nonrelational ones, support context-based authorization policies and access control.

4.4 Custom authorizations (R4)

Some databases only provide a set of built-in access controls, such as the predefined roles (i.e. reader, editor, publisher, architect and admin) in Neo4j (Borojevic, 2017). Custom, declarative conditions allow for much more flexibility, as they do not restrict access constraints (Bertino and Sandhu, 2005; Bertino *et al.*, 2007) to a predefined set of built-in rules.

The fulfillment of this requirement forms the basis to comprehensively benefit from the solutions for the requirements *R2* and *R3* (Bertino *et al.*, 2007; Kulkarni, 2013; Colombo and Ferrari, 2019).

4.5 Separation of concerns (R5)

Our last requirement relates more to the system architecture than to the definition and enforcement of authorizations. Separation of concerns, an important design principle in software development, is also relevant to access control. Handling authorizations and their enforcement within the application is not scalable and hard to maintain. Therefore, access control should be independent of the application and the underlying datastore.

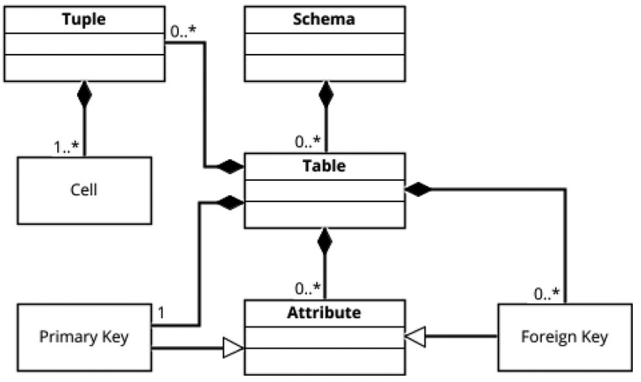
The established policy language and access control model *eXtensible Access Control Markup Language (XACML)* [2], for example, supports separation of concerns. In addition, in research prototypes like the ones presented by Bertino *et al.* (2001), Colombo and Ferrari (2017), Kacimi and Benhlima (2017), Bogaerts *et al.* (2017) and Mohamed *et al.* (2022), authorization policy management and enforcement are also independent of the application and the underlying datastore. However, there are still many integrated and platform-dependent approaches.

The identified requirements are not only implemented differently, but also the definition could vary from one database model to another due to the nature of the data stored in these systems, query language and structure of the data model. For instance, the data model hierarchy, granularity level and resource context are not the same for all database models.

We discuss the proposed requirements in the context of each database model, along with the existing access control features in the upcoming sections.

5. Relational databases

The first relational database management system (RDBMS) evolved in the 1970s and is based on the rigid scientific fundamentals of the relational data model developed by Edgar F. Codd (Zugaj and Beichler, 2019). Figure 1 shows the relational model hierarchy. The data is stored in tables, also referred to as relations. Columns and rows of the table are called attributes and tuples, respectively. Each row has its own unique key. The foreign key is the primary key of another table to link rows in different tables. The relation has a schema (i.e. metadata), which defines the attribute names along with their data type in addition to the instances representing the tuples at a given instant. The RDBMS relies on static schemas to



Source: Authors' own work

Figure 1.
Relational database model

maintain data integrity. Furthermore, relational datastores support only structured data and are managed using the common declarative query language *SQL*.

Access control approaches have been developed in the relational systems ever since the first products emerged. In SQL'89, *discretionary access control (DAC)* is applied such that the relation creator in an SQL database becomes its owner. However, SQL'89 lacks control over who can create relations. The owner can give access privileges (i.e. *select*, *insert* and *delete*) to other users using the grant operation in SQL, which applies to base relations and views. The *drop* relation privilege is not supported in SQL'89, but included in IBM DB2. In addition, the missing revoke operation is provided in SQL'92 with the option of cascading revocation. In general, DAC is prone to *trojan horse* attacks, even if the relation access is strictly controlled. For instance, a user with a *select* privilege can violate these controls by creating a copy of the relation. Another problem is related to access rights management, as the privileges for performing a particular task have to be granted explicitly to each user or group of users (Sandhu, 1994). *Mandatory access control (MAC)* is based on security labels associated with each user or data item. In relational databases, security classifications can be assigned to data at different levels of granularity. In coarse-grained access control, labels are assigned to entire relations or columns, whereas fine-grained refers to the level of tuples or elements. Nevertheless, secret data can be leaked using devious means of communication, i.e. covert channels (Sandhu, 1994).

5.1 Database management system features

Concerning access control in commercial relational database systems, it relies either on creating policy-compliant views and modifying queries to reference them (e.g. MySQL and MariaDB), or using proprietary enforcement mechanisms (e.g. Oracle, PostgreSQL and IBM) (Bao and Clavel, 2021). For instance, MySQL supports row-level access control by creating an abstraction of views and optionally functions to limit users' access by a specific row filtering (Oracle, 2023). On the other hand, the *Oracle Virtual Private Database* (Browder and Davidson, 2002) enforces access control at the row level by appending the expressed content- and context-based conditions in the authorization policy to the where clause of the SQL query.

5.2 Research works

Bertino *et al.* (1997) proposed an access control model for relational databases, supporting permission delegation and negative authorization. Regarding the content-based fine-grained access control in requirement *R2*, there are two categories of enforcement mechanisms: view-based and query rewriting (Colombo and Ferrari, 2016a). In view-based mechanisms, the views are derived according to the specified authorization policy. Access is granted to these views, rather than to the original data resource. On the other hand, the query rewriting enforcement approach intercepts the query to apply the specified constraints in the authorization policy.

A FGAC at the cell level was introduced by LeFevre *et al.* (2004), using dynamically generated views nullifying the unauthorized cells. Moreover, Agrawal *et al.* (2005) proposed a language supporting grant command specification at the cell level (Colombo and Ferrari, 2016a). Research to enhance access control in the RDBMS is still ongoing. For instance, a recent approach to enforce FGAC policies when executing SQL queries is proposed by Bao and Clavel (2021). They presented a generated SQL stored-procedure executing user queries according to fine-grained policies modeled in SecureUML (Lodderstedt *et al.*, 2002). However, an unavoidable performance overhead was introduced at run-time due to the policy complexity.

5.3 Summary

Although existing authorization and access control approaches satisfy our requirements and are advanced, compared to other database models, there are still problems concerning the information revealed when solving subqueries, where-clauses and on-clauses in the devised queries. Furthermore, current access control solutions in the RDBMS are inefficient, error-prone and scale poorly (Bao and Clavel, 2021). As the performance of relational databases degrades with joins, locks and impedance mismatch, nonrelational databases have emerged with various data storage models to address these limitations and handle large amounts of data (Dindoliwala and Morena, 2017).

6. Key-value databases

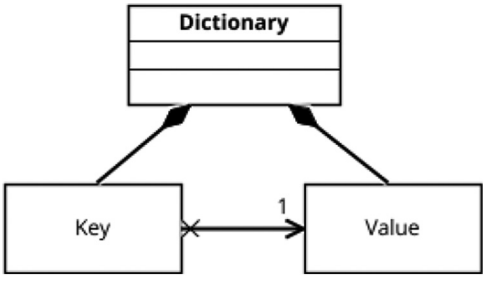
The key-value model uses a hash table and is the simplest one among all NoSQL models. It is powerful and efficient in storing schema-less data in the form of data values associated to keys, which are used as indexes for quickly finding values in large data sets (Alotaibi *et al.*, 2019). Data can be either stored as rows, like structured data, or JSON objects. *Redis [3]* and *Accumulo [4]* are examples of native key-value databases.

Redis is an advanced open-source datastore, where each key-value is a pair of binary strings for managing different types of binary data (e.g. XML documents, images, arrays and bytes). It provides hashes to store and query the database objects. In Accumulo, data is stored in a *distributed sorted map*. The keys are logically divided into a row key to uniquely identify the row, a column and an automatically generated timestamp used for versioning (Moreno *et al.*, 2018). Each column is further divided into a family (i.e. the logical grouping of the key), a qualifier as a more specific key attribute, and a visibility tag, which stores a logical combination of security labels.

According to the key-value model represented in Figure 2, the hierarchy consists of a collection of records identified by their unique keys. The finest granularity in this model is the value of a particular key.

6.1 Database management system features

The Accumulo database applies FGAC at the cell level. Security labels are assigned to key-value pairs by adding an element to the key called *column visibility*. If these labels are satisfied at query time, the respective key and value are included in the response of the user request. On the contrary, Redis can restrict access to specific commands and keys. Role-based access control (RBAC) is supported only in the *Redis Enterprise Cloud* service, but permissions are assigned to the users directly in the open-source version (Gupta *et al.*, 2023).



Source: Authors' own work

Figure 2.
Key-value database model

6.2 Research works

Moreno *et al.* (2018) proposed a model to describe who can access the values of specific data cells in a key-value database system. The labels may define rules for access matrix, RBAC or multilevel models. In this work, enforcement of the specified authorizations was not addressed. There are also proposals to use key-value datastores to manage authorization policies. For example, Colombo and Ferrari (2018) implemented an access control enforcement mechanism on top of key-value databases. However, this work did not conceptually address authorization and access control in the database model.

6.3 Summary

When matching the requirements in Section 4 with the available access control features, we find that the requirement *R1* is satisfied because the model hierarchy is simple (i.e. a table and key-value entries). Requirement *R2* is not satisfied for this model, as the access to the fine-grained element is based on its security label or the assigned role, rather than content. We consider the key-value database model satisfying requirement *R4*, because at least one database system has the option to specify custom authorization policies. For instance, custom security labels can be specified in Accumulo using the column visibility element of the key. Finally, no demonstration cases have been presented for context-based policy (*R3*) and external authorization (*R5*) in key-value databases so far.

7. Column-oriented databases

The column-oriented database is also referred to as column-family store. It is considered as an evolution of key-value datastore, where data is also represented as hash maps, but with more than one indexing level (Abadi *et al.*, 2009). The meta model for column-oriented databases is illustrated in Figure 3. Each column consists of a key and a value. The column-family is a set of rows equivalent to a table in the relational databases. A set of column-family is defined as a key space. This model is typically used in data mining and web applications because of its ability to deal with massive data and complex datasets in distributed systems (Nayak *et al.*, 2013). However, it is less flexible than key-value and document-based models, because the column-family needs a schema at the application level. Examples of column-oriented databases are *Cassandra [5]* and *HBase [6]*.

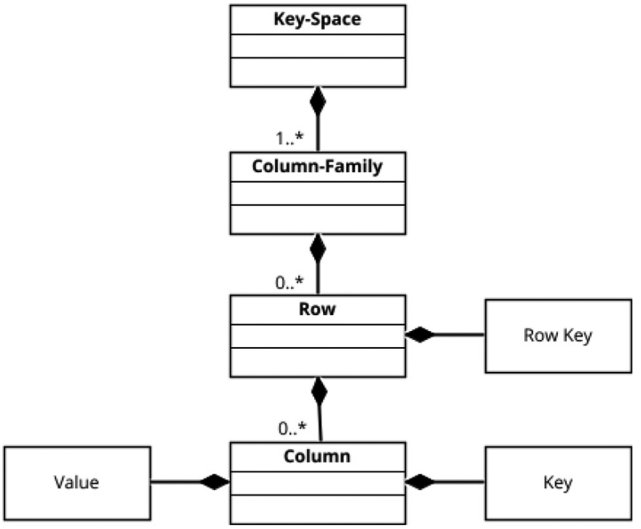
According to the column-oriented database model, authorization policies should specify constraints at the level of key space, column-family, row and column. FGAC should allow to limit access to specific column values within a row.

7.1 Database management system features

Cassandra supports RBAC at the key space or column-family level, according to the user's role(s) and privileges. It uses the grant/revoke security paradigm to manage permissions on resources, which are assigned to roles (Dindoliwala and Morena, 2017). The resource could be key space, role, table, index or function. In Cassandra, access control at the object level is not available (Dadapeer and Adarsh, 2016). HBase enforces authorization using access control lists.

7.2 Research works

Kulkarni (2013) proposed a fine-grained key-value access control model, where authorization policies can be specified at the level of column-family, key space, column or row. However, this model was implemented as a library and restricted to Cassandra and HBase. Shalabi and Gudes (2017) presented a schema for cryptographic enforcement of RBAC policies in



Source: Authors' own work

Figure 3.
Column-oriented database model

Cassandra. This approach is platform-specific and the enforcement overhead is not discussed (Colombo and Ferrari, 2019). In addition, fine-grained, content-based and context-aware policies are not supported.

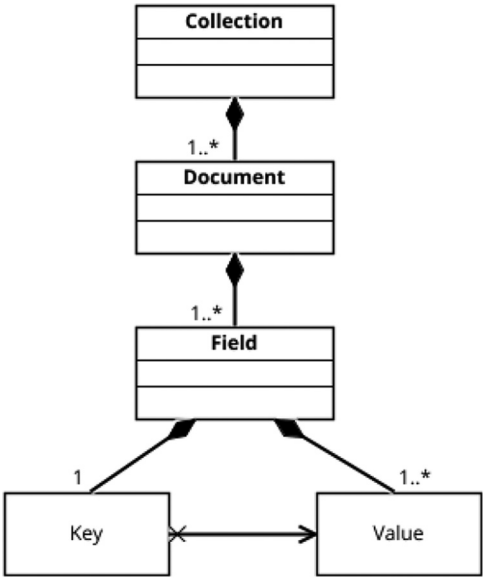
7.3 Summary

Column databases allow to specify custom authorization policies at different levels (*R1* and *R4*). In addition, the work in Kulkarni (2013) claimed to enforce content-based FGAC (*R2*). It also provided three examples to apply context-based access control (*R3*), by describing user location and time of the day in the authorization policy. Nevertheless, the proposed model was implemented on a small-sized patient information system and has not been changed over the last 10 years [7]. Because existing policy languages have no direct support for column-oriented data structures, this database model fails to meet the requirement *R5*.

8. Document-based databases

The document-based database model is the most commonly used NoSQL model, as it can manage structured, semi-structured and unstructured data. Data is stored as schema-less documents with one or more fields in key-value pairs or nested documents (see Figure 4). Documents are analogous to records in the RDBMS and collections are equivalent to tables, but without a predefined schema. Each document is identified by a unique key, which is used to manipulate (i.e. insert, delete and update) document data and for linking documents. For fast data retrieval, fields can be indexed. Document datastores are typically used for blog software and content management systems, due to their flexibility, high performance and horizontal scalability. Examples are *MongoDB [8]* and *Couchbase [9]*.

MongoDB is a distributed general-purpose database that stores data in the form of BSON documents, without schema definition. It uses collections as an additional organization level to group similar documents and provides its own query language, i.e. *MongoDB Query Language*. MongoDB is the first ranked NoSQL database due to its strengths, including the



Source: Authors' own work

Figure 4.
Document-based database model

support of all indexing techniques in relational databases for data sorting and faster searching (Sicari *et al.*, 2022). Besides the pure document-based datastores, there are multimodel database systems relying on the document structure in its core. For instance, ArangoDB is a multimodel database, storing data as documents, which are organized in collections and databases.

The access control model should allow the specification of authorization policies at the level of database, collection, document and field. Database/Collection level policies regulate access to all documents in a database/collection, whereas document level policies cover the entire set of fields included in a document. The finest granularity in this database model is at the level of document fields. A field can be of a simple datatype, a document (i.e. nested document) or even an array of fields (Colombo and Ferrari, 2015a).

8.1 Database management system features

Document datastores support RBAC. Couchbase implements 46 predefined roles with specific privileges on the entire collection [10]. Most of these roles are exclusive to the enterprise version, while only three roles can be used in the community version. MongoDB has built-in and user-defined roles that grant privileges for actions on a resource (i.e. database, collection, set of collections or cluster). Users have no access to the system if they are not assigned to at least one role. The first user created in the database should be a user administrator with privileges to manage other users [11]. Although MongoDB is adopted in many solutions due to its dynamic structure, there is no standardization of authorization and access control. In ArangoDB, external authorization can be applied using *Foxx* microservices [12], which allow enforcing fine-grained permissions directly on the database using APIs.

8.2 Research works

In Colombo and Ferrari (2015b), the RBAC model in MongoDB was enhanced to support purpose-based policy specification at the document level. The same authors presented a research roadmap for integrating context-aware fine-grained access control features into MongoDB (Colombo and Ferrari, 2015a). Subsequently, they refined the granularity level of access control in MongoDB to support content- and context-based policies at the field level (Colombo and Ferrari, 2016b). However, the proposed approaches are limited to MongoDB. They eventually generalized the concept to enforce ABAC into document datastores at the document or field level without prior knowledge of the document structure (Colombo and Ferrari, 2017). This is achieved by introducing SQL++ as a unifying query language for NoSQL, with a query rewriting step to apply fine-grained authorizations. Nevertheless, SQL++ is not widely used in practice, as it needs adaptations to work with the different languages and data models of NoSQL databases (Gupta *et al.*, 2023).

Kacimi and Benhlima (2017) presented an architecture applied to MongoDB for purpose-based access control policies in XACML. It uses a policy language to express the authorization policy, but the enforcement is application-specific. Although they provided an implemented access control as a service approach using XACML with MongoDB, further implementation is needed to map the defined policy attributes to database values. Gupta *et al.* (2023) introduced a framework to specify and enforce field-level permissions. Their concept was based on intercepting a proxy server, which uses the *MongoDB wire protocol* (MongoDB, 2023) and communicates with the database server on behalf of the clients.

8.3 Summary

According to the existing access control features for the document-based database model, we can say that at least one research work addressed each of our requirements. However, the currently proposed approaches still have flaws and need further investigation to be used in practice. Current document datastores support custom authorizations (i.e. roles) (*R4*). Moreover, we consider separation of concerns (*R5*) as a supported feature in the document-based database model, because there are database systems (e.g. the multimodel datastore *ArangoDB*) having a framework to specify and enforce authorization policies externally. However, such frameworks are specific to a particular datastore and cannot be reused in others, even with the same database model. Because most document-based datastores support coarse-grained RBAC, the granularity (*R1*), content (*R2*) and context (*R3*) requirements are not met.

9. Graph databases

The data is stored in graphs as object nodes (vertices) connected by relationships (edges). A graph database has no predefined schema and can be seen as a special kind of document-based database, where both nodes and relationships are represented by documents (Sicari *et al.*, 2022). It is scalable and uses shortest path algorithms for improving the efficiency of data queries, but is more complex to manage. There are different graph models, but the *property graph* (see Figure 5) is the most common model in graph databases.

Graph databases are mainly used in recommendation systems and social networks; however, there is no standard query language for manipulating data and traversing graphs yet. Neo4j (2023) is the top ranked native graph database [13] with its declarative query language *Cypher*.

FGAC in the context of graph-structured data is to protect vertices and edges, along with their properties (i.e. attributes). However, one of the key problems is how to describe the object of a permission (Kalajainen, 2007). Vertices and edges are not isolated and even

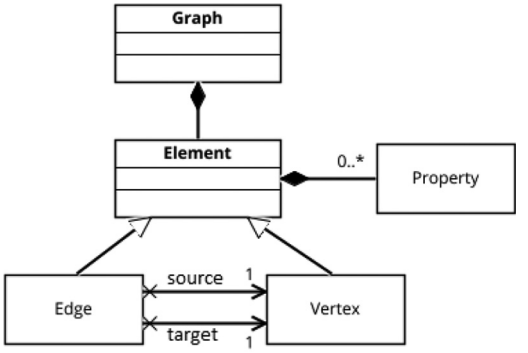


Figure 5.
Property graph database model

Source: Authors' own work

contexts in the graph model can be an object of a permission. Thus, it is not sufficient to protect a single element, but to consider its context, i.e. the path (or parts of it) to the protected element. There could be several paths from a subject to a resource, but not all of them are authorized. In this case, the subject can be denied from accessing this resource because of an unauthorized path. Therefore, the context of resources should be considered when specifying and enforcing authorizations.

9.1 Database management system features

Currently, native (e.g. Neo4j) and non-native (e.g. Microsoft Azure CosmosDB [Weiss *et al.*, 2022] and ArangoDB [Oasis, 2019]) graph databases provide access control based on roles. Neo4j supports RBAC with predefined roles (i.e. reader, editor, publisher, architect and admin), in addition to subgraph and property-level access control (Borojevic, 2017). It also has a special database (i.e. system database) for storing the privileges. Regarding the multimodel database systems supporting graph structures, access control features apply the same way like in other models.

9.2 Research works

Morgado *et al.* (2018) introduced a model-based approach using metadata with authorization rules to control access in applications that use graph databases. It provides a predefined schema for the graph nodes in addition to data definition language and data manipulation language operations. This model only allows specifying positive permissions on individual nodes. The work did not show how the model handles conflicts. In Valzelli *et al.* (2021), the authors proposed an initial solution to protect knowledge graphs. A knowledge graph contains the main entities in a certain domain, along with their relations. They introduced a property graph model to specify open and closed policies, using authorization edges between subjects (i.e. user and user group) and resources (or resource category). However, they focused only on DAC, MAC and RBAC, which cannot enforce FGAC. Both papers provided conceptual approaches that need to be implemented on top of the graph model.

Relationship-based access control (ReBAC) and attribute-based access control (ABAC) are examples of content-based models. ReBAC is an access control model based on evaluating relationships between accessing subjects and target resources, e.g. the *friendship* relation in social network applications. ReBAC approaches are either based on hybrid logic,

like Bruns *et al.* (2012) and Fong *et al.* (2013), or path regular expressions on edges (e.g. type or depth), like Crampton and Sellwood (2014). During the past decade, different ReBAC models have emerged, e.g. Giunchiglia *et al.* (2008), Fong (2011) and Cheng *et al.* (2016). Clark *et al.* (2022) compared representative ReBAC models and derive ReBAC policy language features, which are formalized in their ReBAC query language *ReLOG*. They considered ReBAC policies as graph queries over graph databases. Currently, there is no common definition of ReBAC, but a number of domain-specific models with rather ad-hoc enforcement models and implementations. However, ReBAC generally does not support specification of fine-grained constraints on the protected entities.

On the contrary, there is ongoing research based on the ABAC model, to define and apply fine-grained constraints in the authorization policy on the level of vertex and edge attributes. For instance, Bertolissi *et al.* (2019) presented an access control policy framework extending the ABAC model with the notion of path condition. This is because the ABAC model does not consider the specific relationships that might exist between subjects and resources. The approach uses the extensibility points of XACML to add a module for resolving path queries and references the user-defined function in the policy. The mechanism of access constraints resolution is decoupled from the XACML implementation. Thus, the approach requires minimal implementation effort to evaluate path constraints in XACML. A disadvantage of this approach is that the semantics of graph paths – including attributes of vertices and edges, vertex labels and edge types – are encoded within the user-defined function, which could be nested. Thus, the expressiveness of the policy is limited, as only the function is referenced, while the path-related constraints are not specified in the policy. This implies modifying the user-defined functions upon changing the policy. In addition, each edge type, subpaths and custom conditions require a new function.

A recent work by Hofer *et al.* (2022) contributed to enhancing access control in object graph mappers to support FGAC at the level of attributes. This is done by intercepting the communication between the database and the mapper framework, providing a central authorization point where additional filters can be applied. The authors highlighted the need for a general fine-grained approach that is implementation-independent. Mohamed *et al.* (2021) coupled ABAC with a new declarative language for fine-grained, attribute-based authorization policy, named *XACML for Graph-structured data (XACMLAG)*. The same authors proposed an extension to the XACML language and architecture in Mohamed *et al.* (2022) to specify and enforce fine-grained constraints on graph paths. Even though additional path-specific constraints in terms of graph patterns can be described, the policy rules require specialized processing and the enforcement mechanism needs to be adapted to work in a specific graph datastore (Sicari *et al.*, 2022).

9.3 Summary

Current access control features in graph databases do not meet requirement *R1*, but there are several recent research works toward enhancing ABAC to support relationships and ReBAC to apply FGAC. The RBAC privileges in Neo4j are limited to reading/updating the database, managing resources (i.e. databases, users, roles and privileges) as well as editing node labels, relationship types and property names. To the best of our knowledge, content-based FGAC cannot be applied in the existing graph databases (*R2*). The achieved authorization in the enterprise edition of Neo4j specifies privileges using static commands in terms of actions to be performed (e.g. traverse, read and match) on particular node labels or relationship types within graphs. These privileges are then granted (or denied) to roles. The proposed XACML-based approaches are examples to apply content-based access control

using the ABAC model for graph-structured data. The other works, relying on the ReBAC model, also consider content-based authorizations. Existing graph databases and literature research including the policy language *XACML4G* did not show rule specification and enforcement taking user context information, e.g. access time and location of the user, into account (*R3*). Custom authorizations (*R4*) are supported in graph DBMS and literature. For instance, custom roles can be defined in the enterprise edition of Neo4j. According to the research works presented earlier in this section, custom attributes and user-defined functions can be added. Finally, separation of concerns (*R5*) is satisfied, because authorization policies can be specified and enforced in the database layer (e.g. system database in Neo4j) as well as externally (e.g. *Foxx* microservices in ArangoDB). Furthermore, there are recent research works addressing this requirement, such as Bertolissi *et al.* (2019) and Mohamed *et al.* (2021). However, these approaches are datastore-specific.

10. Discussion

In the previous sections, we provided the current state of the art of authorization and access control features, supported within database systems or introduced by research works in relational as well as NoSQL database models. We now relate the requirements defined in Section 4 with the previously discussed features, taking into consideration that the data model hierarchy is structured differently and the term *fine-grained* is defined differently for each database model.

To address *RQ3*, we summarize the assessment of the requirements in Table 1. For each database model, we indicate whether a requirement is satisfied in at least one DBMS and research work with a ✓, addressed in the literature only with a o^l, or not at all with a ×. Features supported in a NoSQL database or addressed in a research do not necessarily apply to other datastores of the same database model. Furthermore, approaches proposed in research works should be studied in detail before being applied in practice as they can be still work in progress, need further evaluation, have an outdated implementation or are even limited to a specific datastore/application.

The relational model has the most sophisticated authorization and access control mechanisms in comparison to all NoSQL models. However, it is not scalable to deal with big interconnected data. The ABAC model can be enforced in relational databases to apply FGAC based on content taking environmental conditions into account. Moreover, custom rules can be specified at different levels within the database (e.g. using views) or externally using a policy language, such as XACML. However, there are still performance, scalability and information leakage limitations.

NoSQL databases trade consistency and security for performance and scalability (Dindoliwala and Morena, 2017). The access control approaches available in the literature are specific to certain database models or even datastores. This is due to the lack of a reference model. Furthermore, the same database model can have multiple implementations (Colombo and Ferrari, 2016a).

Table 1. Satisfied requirements for each database model

Database model	R1	R2	R3	R4	R5
Relational	✓	✓	✓	✓	✓
Key-value	✓	×	×	✓	×
Column-oriented	o^l	o^l	o^l	✓	×
Document-based	o^l	o^l	o^l	✓	✓
Graph	o^l	o^l	×	✓	✓

The key-value model is the simplest in terms of structure and hierarchy; however, it has the least information security support. For now, only basic authorization using labeling or roles is supported. Current research works focus on enforcing policies using key-value databases, rather than enhancing their authorization and access control features (e.g. content-based, context-based and separation of concerns).

For the column-oriented database model, only custom authorization is supported in some datastores (e.g. roles in Cassandra). The only research work to enhance access control for this model addresses fine-grained, content-based and context-aware access control. Nevertheless, further investigation is needed in terms of applicability in real cases and deprecation of source code.

The document-structured database model received the most attention among the NoSQL ones. The access control features of datastores, together with the research works, addressed our authorization and access control requirements. Pure and multimodel document datastores only provide coarse-grained RBAC, but custom roles can be created in some DBMS products (e.g. MongoDB). There are many research works addressing the rest of our requirements. However, a general approach satisfying these advanced requirements is still missing for the document-based model. Therefore, research-based solutions have to be carefully selected not only according to the requirements of the access control system, but also while considering their limitations.

Regarding access control for graph databases, Neo4j is more advanced than other graph databases, e.g. custom roles and separate permission management. However, existing access control features still need to be enhanced to not only provide access control for nodes and relationships based on content (e.g. attributes), but also for protecting the graph while traversing it. Accordingly, recent research works focus on fine-grained authorization policy specification and enforcement using the ReBAC and ABAC models. The proposed approaches have three main drawbacks:

(1) not generally applicable because graph datastores have different query languages;

(2) extra implementation is required upon changing or adding new policies; and

(3) context information is not considered.

It is still challenging to meet the missing requirements and come up with an authorization policy language and enforcement model that fits within one or more nonrelational database models. There is no common query language and database features are not consistently supported in the different datastores, even those relying on the same database model. This makes it hard to switch from one NoSQL datastore to another. Thus, an access control solution that applies to all database models is still missing.

11. Conclusion

NoSQL database models, such as key-value, document-based, column-oriented and graph, found their way into practice in addition to the relational one. The focus of NoSQL models has not been on security, but rather on efficiency in processing huge amounts of data or traversing heavily interconnected ones. However, they are now used in security-relevant areas, which demand sophisticated data protection.

Access control ensures information security and protection by enforcing authorizations, in terms of which users are permitted (or denied) to perform what operations on which resources. For the relational model, robust access control approaches and mechanisms to protect sensitive information are established in the current RDBMS. However, RDBMS are

inefficient in storing and handling big data. For the NoSQL models, authorization and access control are still challenging.

In the traditional relational model, data is structured into tables with a fixed schema, where each data entry is equivalent to a row having values for the columns. The NoSQL models have different forms. First, the key-value model is represented as a hash table with key-value entries. In the column-oriented model, records hold a collection of dynamic columns that are grouped into column families within a key space. The document-based model consists of collections having document entries in the form of key-value pairs or nested documents, whereas the graph data structure is solely based on vertices and edges.

In this paper, we address authorization and access control, with respect to requirements and features within existing database systems or research works, for all these database models. We start our research with a SLR on survey works addressing authorization and access control to answer *RQ1*. None of the works only focus on authorization and access control in detail, but rather on security in databases (mostly NoSQL databases) or data security in general (cp. Section 3).

In our systematic literature research, we additionally focus on the identification of requirements to generally apply fine-grained dynamic authorization and access control for different database models, regardless of the application scenario (see *RQ2*). We identified five requirements: (*R1*) varying granularity of protected data objects with focus on fine-grained authorizations, (*R2*) content-based, (*R3*) context-based, (*R4*) custom authorizations and (*R5*) separation of concerns. Due to the different data models and definitions of FGAC, we discuss each of the data models in general and with respect to our authorization and access control requirements to answer *RQ3*. We provide an overview of the current state of the art in research and practice for each of the models using sources about database products as well as research papers. Finally, we match the features of the discussed models and products with our defined requirements in the summary Table 1 in Section 10.

All of our requirements are currently only met by the relational data model and its implementations, but with drawbacks in performance, scalability and vulnerability in revealing information used to reach the final result of the executed query. Customized authorizations and their enforcement (*R4*) are available with the NoSQL models. FGAC (*R1*) is also supported with key-value datastores in practice, however in research, they are discussed with the column-oriented, document-based and graph models. Content-based (*R2*) and context-based authorizations (*R3*) are not considered with NoSQL products so far. However, content-based authorizations are intensively studied in the research community with column-oriented, document-based and graph models. Context-based authorizations for NoSQL datastores are not the main focus in the current research. We only found contributions in the context of column-oriented and document-based models. Separation of concerns (*R5*) is currently not a priority in the context of access control for NoSQL datastores. ArangoDB considers external authorizations on the product side and only a few research results are dealing with external authorizations, policy languages as well as access control frameworks and architectures for NoSQL models.

Still more research is needed on authorization and access control for the NoSQL models to achieve a maturity level similar to that in the relational one. A lot of research and development is already focusing on more sophisticated authorization and access control for specific NoSQL platforms, but also platform-independent approaches such as common query languages like SQL++. However, Colombo and Ferrari (2019) emphasized that a unifying access control framework for NoSQL datastores is still missing. They considered the heterogenous schemaless data in the key-value, column and document databases as the main reason. Thus, they proposed a tree-structured meta-model as a basis for implementing

the general access control mechanism. From our perspective, graph and relational models should also be considered in such a unifying approach. As graphs have a more general structure than trees, we will continue our work on authorization and access control for graph-structured data, focusing on platform independence and performance. In addition, research on a graph-structured meta-model integrating several other NoSQL models seems promising.

Notes

1. www.citavi.com/en
2. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html
3. https://redis.io
4. https://accumulo.apache.org/
5. https://cassandra.apache.org
6. https://hbase.apache.org/
7. https://github.com/devdattakulkarni/KVAC
8. www.mongodb.com/
9. www.couchbase.com/
10. https://docs.couchbase.com/server/current/rest-api/rbac.html
11. www.mongodb.com/docs/manual/core/authorization
12. www.arangodb.com/docs/stable/foxx.html
13. https://db-engines.com/en/ranking/graph+dbms

References

Abadi, D.J., Boncz, P.A. and Harizopoulos, S. (2009), "Column-oriented database systems", *Proceedings of the VLDB Endowment*, Vol. 2 No. 2, pp. 1664-1665, doi: 10.14778/1687553.1687625. issn: 2150-8097.

Agrawal, R., Bird, P., Grandison, T., Kiernan, J., Logan, S. and Rjaibi, W. (2005), "Extending relational database systems to automatically enforce privacy policies", *21st International Conference on Data Engineering (ICDE'05)*, pp. 1013-1022, doi: 10.1109/ICDE.2005.64.

Alotaibi, A., Alotaibi, R. and Hamza, N. (2019), "Access control models in NoSQL databases: an overview", *Journal of King Abdulaziz University (JKAU)*, Vol. 8 No. 1, pp. 1-9.

Bacon, J., Moody, K. and Yao, W. (2003), "Access control and trust in the use of widely distributed services", *Software: Practice and Experience*, Vol. 33 No. 4, pp. 375-394, doi: 10.1002/spe.511.

Bao, H.N.P. and Clavel, M. (2021), "A model-driven approach for enforcing fine-grained access control for SQL queries", *SN Computer Science*, Vol. 2 No. 5, p. 370, doi: 10.1007/s42979-021-00712-7.

Bertino, E. and Sandhu, R. (2005), "Database security - concepts, approaches, and challenges", *IEEE Transactions on Dependable and Secure Computing*, Vol. 2 No. 1, pp. 2-19, doi: 10.1109/TDSC.2005.9.

Bertino, E., Byun, J.-W. and Kamra, A. (2007), "Database security", in Petković, M. and Jonker, W. (Eds), *Security, Privacy, and Trust in Modern Data Management*, Data-Centric Systems and Applications, Springer-Verlag, Berlin, Heidelberg, pp. 87-101. isbn: 978-3-540-69860-9, doi: 10.1007/978-3-540-69861-6_7.

Bertino, E., Castano, S. and Ferrari, E. (2001), "Securing XML documents with Author-X", *IEEE Internet Computing*, Vol. 5 No. 3, pp. 21-31, doi: 10.1109/4236.935172. issn: 10897801.

Bertino, E., Ghinita, G. and Kamra, A. (2011), *Access Control for Databases: Concepts and Systems*, Now Publishers.

Bertino, E., Samarati, P. and Jajodia, S. (1997), "An extended authorization model for relational databases", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 9 No. 1, pp. 85-101, doi: 10.1109/69.567051. issn: 10414347.

Bertolissi, C., den Hartog, J. and Zannone, N. (2019), "Using provenance for secure data fusion in cooperative systems", *Proceedings of the 24th ACM Symposium on Access Control Models and Technologies. SACMAT '19, Association for Computing Machinery, Toronto ON, Canada*, pp. 185-194, isbn: 9781450367530, doi: 10.1145/3322431.3325100.

Bogaerts, J., Lagaisse, B. and Joosen, W. (2017), "SEQUOIA: scalable policy-based access control for search operations in data-driven applications", in Bodden, E., Payer, M. and Athanasopoulos, E. (Eds), *Engineering Secure Software and Systems*, Springer International Publishing, Cham, pp. 1-18. isbn: 978-3-319-62105-0, doi: 10.1007/978-3-319-62105-0_1.

Borojevic, I. (2017), "Role-based access control in Neo4j enterprise edition", available at: https://neo4j.com/blog/role-based-access-control-neo4j-enterprise (accessed April 2023).

Browder, K. and Davidson, M.A. (2002), "The virtual private database in oracle9ir2", *Oracle Technical White Paper, Oracle Corporation*, Vol. 500 No. 280.

Bruns, G., Fong, P.W., Siahaan, I. and Huth, M. (2012), "Relationship-based access control: its expression and enforcement through hybrid logic", *Proceedings of the Second ACM Conference on Data and Application Security and Privacy. CODASPY '12*, Association for Computing Machinery, San Antonio, TX, pp. 117-124, isbn: 9781450310918, doi: 10.1145/2133601.2133616.

Cheng, Y., Park, J. and Sandhu, R. (2016), "An access control model for online social networks using user-to-user relationships", *IEEE Transactions on Dependable and Secure Computing*, Vol. 13 No. 4, pp. 424-436, doi: 10.1109/TDSC.2015.2406705.

Clark, S., Yakovets, N., Fletcher, G. and Zannone, N. (2022), "ReLOG: a unified framework for relationship-based access control over graph databases", *Proceedings of Data and Applications Security and Privacy XXXVI: 36th Annual IFIP WG 11.3 Conference, DBSec 2022, Newark, NJ, USA*, July 18–20, 2022, Springer-Verlag, Newark, NJ, USA, pp. 303-315, isbn: 978-3-031-10683-5, doi: 10.1007/978-3-031-10684-2_17.

Colombo, P. and Ferrari, E. (2018), "Access control enforcement within MQTT based internet of things ecosystems", *Proceedings of the 23nd ACM on Symposium on Access Control Models and Technologies. SACMAT '18, Association for Computing Machinery, Indianapolis, IN*, pp. 223-234, isbn: 9781450356664, doi: 10.1145/3205977.3205986.

Colombo, P. and Ferrari, E. (2019), "Access control technologies for big data management systems: literature review and future trends", *Cybersecurity*, Vol. 2 No. 1, pp. 1-13.

Colombo, P. and Ferrari, E. (2015a), "Enhancing MongoDB with fine grained context-aware access control", *3rd International IBM Cloud Academy Conference - ICACON 2015*, p. 7.

Colombo, P. and Ferrari, E. (2015b), "Enhancing MongoDB with purpose-based access control", *IEEE Transactions on Dependable and Secure Computing*, Vol. 14 No. 6, pp. 591-604, doi: 10.1109/TDSC.2015.2497680.

Colombo, P. and Ferrari, E. (2016a), "Fine-grained access control within NoSQL document-oriented datastores", *Data Science and Engineering*, Vol. 1 No. 3, pp. 127-138, doi: 10.1007/s41019-016-0015-z.

Colombo, P. and Ferrari, E. (2017), "Towards a unifying attribute based access control approach for NoSQL datastores", *2017 IEEE 33rd International Conference on Data Engineering (ICDE), IEEE*, pp. 709-720, isbn: 978-1-5090-6543-1, doi: 10.1109/ICDE.2017.123.

Colombo, P. and Ferrari, E. (2016b), "Towards virtual private NoSQL datastores", *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pp. 193-204, doi: 10.1109/ICDE.2016.7498240.

Crampton, J. and Sellwood, J. (2014), "Path conditions and principal matching: a new approach to access control", *Proceedings of the 19th ACM Symposium on Access Control Models and Technologies. SACMAT '14, Association for Computing Machinery, London, Ontario, Canada*, pp. 187-198, isbn: 9781450329392, doi: 10.1145/2613087.2613094.

Dadapeer, N.I. and Adarsh, G. (2016), "A survey on security of NoSQL databases", *International Journal of Innovative Research in Computer and Communication Engineering*, Vol. 4 No. 4.

Dindoliwala, V.J. and Morena, R.D. (2017), "Survey on security mechanisms in NoSQL databases", *International Journal of Advanced Research in CS*, Vol. 8 No. 5.

Ferrari, E. (2009), "Database security", in Ozsu, M.T. and Liu, L. (Eds), *Encyclopedia of Database Systems*, Springer Reference, Springer, New York, NY, pp. 728-732. isbn: 978-0-387-35544-3, doi: 10.1007/978-_0-_387-_39940-9_111, available at: https://rdcu.be/c9PLH

Fong, P.W. (2011), "Relationship-based access control: protection model and policy language", *Proceedings of the First ACM Conference on Data and Application Security and Privacy. CODASPY '11, Association for Computing Machinery, San Antonio, TX, USA*, pp. 191-202, isbn: 9781450304665, doi: 10.1145/1943513.1943539.

Fong, P.W., Mehregan, P. and Krishnan, R. (2013), "Relational abstraction in community-based secure collaboration", *Proceedings of the 2013 ACM SIGSAC Conference on Computer Communications Security. CCS '13, Association for Computing Machinery Berlin, Germany*, pp. 585-598, isbn: 9781450324779, doi: 10.1145/2508859.2516720.

Giunchiglia, F., Zhang, R. and Crispo, B. (2008), "RelBAC: relation based access control", *2008 Fourth International Conference on Semantics, Knowledge and Grid, Beijing, China*, pp. 3-11, doi: 10.1109/SKG.2008.76.

Gupta, E., Sural, S., Vaidya, J. and Atluri, V. (2023), "Enabling attribute-based access control in NoSQL databases", *IEEE Transactions on Emerging Topics in Computing*, Vol. 11 No. 1, pp. 208-223, doi: 10.1109/TETC.2022.3193577.

Hofer, D., Nadschläger, S., Mohamed, A. and Küng, J. (2022), "Extending authorization capabilities of object relational/graph mappers by request manipulation", in Strauss, C., Cuzzocrea, A., Kotsis, G., Tjoa, A.M. and Khalil, I. (Eds), *Database and Expert Systems Applications*, Springer International Publishing, Cham, pp. 71-83. isbn: 978-3-031-12426-6.

Kacimi, Z. and Benhlima, L. (2017), "XACML policies into MongoDB for privacy access control", *Proceedings of the Mediterranean Symposium on Smart City Application. SCAMS '17, Association for Computing Machinery, Tangier, Morocco*, isbn: 9781450352116, doi: 10.1145/3175628.3175

Kalajainen, T. (2007), "An access control model in a semantic data structure: case process modelling of a bleaching line", Department of CS and Engineering.

Kirrane, S. (2015), "Linked data with access control", Dissertation, National University of Ireland, Galway.

Kulkarni, D. (2013), "A fine-grained access control model for Key-Value systems", *Proceedings of the Third ACM Conference on Data and Application Security and Privacy. CODASPY '13, Association for Computing Machinery, San Antonio, TX*, pp. 161-164, isbn: 9781450318907, doi: 10.1145/2435349.2435370.

LeFevre, K., Agrawal, R., Ercegovac, V., Ramakrishnan, R., Xu, Y. and DeWitt, D. (2004), "Limiting disclosure in hippocratic databases", *30th International Conference on Very Large Databases, VLDB Endowment, Toronto, Canada*, pp. 108-119.

Lodderstedt, T., Basin, D. and Doser, J. (2002), "SecureUML: a UML-based modeling language for model-driven security", in Jézéquel, J.-M., Hussmann, H. and Cook, S. (Eds), *≪UML≫ 2002 — the Unified Modeling Language*, Springer, Berlin, Heidelberg, pp. 426-441. isbn: 978-3-540-45800-5.

Mohamed, A., Auer, D., Hofer, D. and Küng, J. (2021), "Extended authorization policy for graph-structured data", *SN Computer Science*, Vol. 2 No. 5, pp. 1-18.

Mohamed, A., Auer, D., Hofer, D. and Küng, J. (2022), "Extended XACML language and architecture for access control in graph-structured data", *The 23rd International Conference on Information Integration and Web Intelligence. iiWAS2021. Association for Computing Machinery, Linz, Austria*, pp. 367-374, isbn: 9781450395564, doi: 10.1145/3487664.3487789.

MongoDB, I. (2023), "MongoDB wire protocol", MongoDB, Inc, available at: www.mongodb.com/docs/manual/reference/mongodb-wire-protocol/ (accessed April 2023).

Moreno, J., Fernandez, E.B., Fernandez-Medina, E. and Serrano, M.A. (2018), "A security pattern for key-value NoSQL database authorization", *Proceedings of the 23rd European Conference on Pattern Languages of Programs. EuroPLoP '18, Association for Computing Machinery, Irsee, Germany*, isbn: 9781450363877, doi: 10.1145/3282308.3282321.

Morgado, C., Baioco, G.B., Basso, T. and Moraes, R. (2018), "A security model for access control in graph-oriented databases", *2018 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, pp. 135-142, doi: 10.1109/QRS.2018.00027.

Nayak, A., Poriya, A. and Poojary, D. (2013), "Type of NOSQL databases and its comparison with relational databases", *International Journal of Applied Information Systems*, Vol. 5 No. 4, pp. 16-19.

Neo4j, I. (2023), "Neo4j documentation", Neo4j, Inc, available at: https://neo4j.com/docs/ (accessed April 2023).

Oasis (2019), "Access control in ArangoDB", available at: www.arangodb.com/docs/stable/oasis/access-control.html (accessed April 2023).

Okman, L., Gal-Oz, N., Gonen, Y., Gudes, E. and Abramov, J. (2011), "Security issues in NoSQL databases", *2011IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications*, IEEE, pp. 541-547, isbn: 978-1-4577-2135-9, doi: 10.1109/TrustCom.2011.70.

Oracle, I. (2023), "MySQL documentation: access control and account management", available at: https://dev.mysql.com/doc/refman/8.0/en/access-control.html (accessed April 2023).

Sahafizadeh, E. and Nematbakhsh, M.A. (2015), "A survey on security issues in big data and NoSQL", *In: Advances in Computer Science: An International Journal*, Vol. 4 No. 4, pp. 68-72.

Samaraweera, G.D. and Chang, J.M. (2021), "Security and privacy implications on database systems in big data era: a survey", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 33 No. 1, pp. 239-258, doi: 10.1109/TKDE.2019.2929794.

Sandhu, R. (1994), "Relational database access controls", *Handbook of Information Security Management*, Auerbach Publications, Boston, 95, pp. 145-160.

Shalabi, Y. and Gudes, E. (2017), "Cryptographically enforced role-based access control for NoSQL distributed databases", in Livraga, G. and Zhu, S. (Eds), *Data and Applications Security and Privacy XXXI*, Springer International Publishing, Cham, pp. 3-19. isbn: 978-3-319-61176-1.

Sicari, S., Rizzardi, A. and Coen-Porisini, A. (2022), "Security & privacy issues and challenges in NoSQL databases", *Computer Networks*, Vol. 206, p. 108828, doi: 10.1016/j.comnet.2022.108828, issn: 1389-1286.

Tankard, C. (2012), "Big data security", *Network Security*, Vol. 2012 No. 7, pp. 5-8, doi: 10.1016/S1353-4858(12)70063-6, issn: 1353-4858.

Valzelli, M., Maurino, A., Palmonari, M. and Spahiu, B. (2021), "Towards an access control model for knowledge graphs".

Weiss, T., Brown, M., Assaf, W., Dale, K., Gunda, S., Sharkey, K., Robbins, M.F., Howell, J., Coulter, D., Lyon, R., Kanshi, G. and Nehme, R. (2022), "Azure role-based access control in azure cosmos DB", Microsoft, available at: https://docs.microsoft.com/en-us/azure/cosmos-db/role-based-access-control (accessed April 2023).

Zahid, A., Masood, R. and Shibli, M.A. (2014), "Security of sharded NoSQL databases: a comparative analysis", *2014 Conference on Information Assurance and Cyber Security (CIACS)*, pp. 1-8, doi: 10.1109/CIACS.2014.6861323.

Zugaj, W. and Beichler, A. (2019), "Analysis of standard security features for selected NoSQL systems", *American Journal of Information Science and Technology*, Vol. 3 No. 2, pp. 41-49.

Corresponding author
Aya Khaled Youssef Sayed Mohamed can be contacted at: aya.mohamed@jku.at