Database Access Control in a Database Keyword Search Tool

Grettel Monteagudo Garcia[1], **Javier Guillot Jiménez**[1],
Angelo Batista Neves Júnior[1], **Yenier Torres Izquierdo**[1],
Melissa Lemos[1], **Marco A. Casanova**[1,2], **Ana Cristina Ferreira**[3],
Flavia Pacheco Texeira da Silva[3]

[1] Tecgraf Institute, PUC-Rio – Rio de Janeiro – RJ – Brazil

[2]Department of Informatics, PUC-Rio – Rio de Janeiro – RJ – Brazil

[3]Petrobras – Rio de Janeiro – RJ – Brazil

{ggarcia, javiergj, angeloneves}@tecgraf.puc-rio.br,

{ytorres, melissa}@tecgraf.puc-rio.br,

casanova@inf.puc-rio.br,

{ana.ferreira, fpacheco}@petrobras.com.br

Abstract. *This paper addresses the access control problem in the context of database keyword search, when a user defines a query by a list of keywords, and not by SQL (or SPARQL) code. It describes the solutions implemented in* DANKE*, a database keyword search platform currently used in several industrial applications.* DANKE *offers two alternatives for managing access control: given a keyword query K and the user permissions P, either compile K into the same structured query and filter the results based on P, or compile K into different structured queries, depending on P. Likewise,* DANKE *has two alternatives for defining the user permissions: using the features of the database management system, or using an internal mechanism.*

1. Introduction

Database access control refers to the problem of restricting access to sensitive data stored in a database only to authorized users. Role-Based Access Control (RBAC) is a popular access control method that grants access to data based on the user's role in the company and enforces the principle of least privilege, meaning that users can only access the data required to perform their job. Any access control mechanism must be supported by an authentication mechanism to ensure proper functioning. An authentication mechanism validates the identity of users and confirms that they are who they claim to be. Without proper authentication, access control mechanisms cannot accurately determine whether a user is authorized to access a particular resource. Regarding database access control, the actions that users can perform over data depend on their permissions, and the data on which these actions are performed can be secured.

In relational databases, row and column access control (RCAC) introduces fine-grained access control as an additional data security layer. Column-Level Access Control allows hiding sensitive columns from users who do not have the necessary privileges, even though they may have access to the underlying table. Row-Level Access Control allows for fine-grained control over which rows of data are visible to different users or roles.

This paper addresses the access control problem in the context of database keyword search, when a user defines a query by a list of keywords, and not by SQL (or SPARQL) code. This scenario poses new challenges, not addressed in the literature, since a database keyword search tool does have, a priori, the list of database tables (or RDF classes and properties) that the keyword query is about, but such list is computed as part of the keyword query compilation process.

The main contribution of the paper is to describe the solutions implemented in DANKE [Izquierdo et al. 2021], a database keyword search platform currently used in several industrial applications. The paper includes two use cases to validate the solution over relational databases, including the industrial application that motivated this investigation.

DANKE offers two alternatives for managing access control. Given a keyword query K and the user permissions P, the first alternative always compiles K into the same structured query, but the tool filters the results based on P. The second alternative compiles K into different structured queries, based on P. Likewise, DANKE has two alternatives for defining the user permissions. In the first alternative, DANKE manages different connection settings, depending on P, and it is up to the database management system to implement access control. In the second alternative, user permissions are defined through metadata associated with DANKE's internal structures.

The remainder of this paper is organized as follows. Section 2 summarizes related work. Section 3 describes the proposed solution in the context of DANKE. Section 4 covers the implementation of access control in DANKE. Section 5 presents two use cases. Finally, Section 6 contains the conclusions and directions for future work.

2. Related Work

This section briefly reviews work in areas that are closely related to the focus of the paper, database security and database keyword search.

The work presented in [Bertino and Sandhu 2005] surveys relevant concepts underlying database security and summarizes the well-known techniques, focusing on access control systems. The key access control models are described, including the discretionary and mandatory access control models and the Role-Based Access Control (RBAC) model.

RBAC [Sandhu et al. 1996, Ferraiolo et al. 2001] is a widely used security paradigm that grants users access to resources based on their role in the organization. Specifically, with RBAC, system administrators create roles according to the organizational functions performed in a given domain, grant permissions to those roles, and then assign users to the roles based on their specific job responsibilities and qualifications [Sandhu et al. 1996]. NIST [Sandhu et al. 2000] is a standard reference model for RBAC that integrates ideas from prior RBAC models, commercial products, and research prototypes.

RBAC has been extended in various ways, such as the Temporal-RBAC (TRBAC) model, introduced by [Bertino et al. 2000], which addresses temporal dependencies between roles. The GEO-RBAC model proposed by [Damiani et al. 2007] extends RBAC with functions that are activated based on user geolocation. Additionally, to support spatial and temporal constraints, the Spatial-Temporal RBAC (STRBAC) model was introduced by [Kumar and Newman 2006].

In [Bertino et al. 2011], a comprehensive overview of models, systems, and approaches proposed for specifying and enforcing access control policies in database management systems is provided, including case studies on fine-grained and context-based access control, mandatory access control, and protection against insider threats. Similarly, [Jabal et al. 2019] presents an extensive review of existing methods, systems, and frameworks for policy analysis across different domains and analysis goals.

As for database keyword search, DANKE [García 2020, Izquierdo et al. 2020] is a keyword search platform that operates over both relational databases and RDF datasets. It is designed to compile a keyword query into a SQL or SPARQL query that returns the best data matches. It works by translating a keyword-based query into a conceptual query and then compiling this conceptual query into a concrete query (either SPARQL or SQL), where each result of the concrete query is an answer to the original keyword-based query. The component explores the database schema to synthesize database queries with projections and selections and joins involving several tables, without user intervention. One of the key features of DANKE is its ability to automatically summarize answers that may be lengthy, enabling users to identify relevant information quickly.

3. The DANKE Platform

Basic architecture. The DANKE architecture comprises three main components: (1) Dataset Preparation; (2) the DANKE Dataset; and (3) Data and Knowledge Extraction. Also, it provides a REST API (Application Programming Interface). Among the available services is the login service (*Logger*), which allows identifying the user that accesses the platform and his credentials.

DANKE features an ABSTRACT SCHEMA, which is a graph that defines a conceptual schema and is independent of the conceptual model of the underlying database (relational or RDF).

The DANKE search process has two main modules, SEARCH and QUERY-PROCESSING, as shown in Figure 1. Given a user keyword query, represented by a list of keywords, the SEARCH module first matches each keyword with labels and values that occur in the database, and then, using the most relevant matches, generates a conceptual query over an abstract schema. The QUERY-PROCESSING module compiles the conceptual query into a structured query over the concrete database schema, executes the structured query, and returns the results to the user. This process does not take into account data access security.

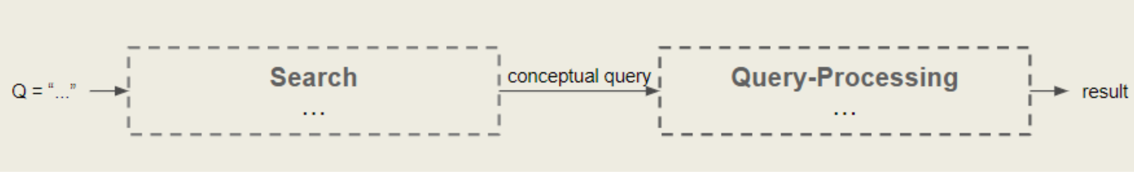


Figure 1. Search Process in DANKE

Access control implementation strategies. DANKE considers two possible strategies to implement access control.

In the *search process result security* strategy, the SEARCH module generates the same conceptual query, regardless of user permissions, but the QUERY-PROCESSING module filters the results (data), based on the user permissions. This implies that for the same set of keywords and two users with different permissions, DANKE generates the same conceptual query, but shows different data, according to the user permissions. DANKE indicates which elements the user does not have access permission.

In the *search process query security* strategy, DANKE synthesizes a different conceptual query for each user, considering the user permissions, in the best possible way. This implies that, for the same set of keywords and two users with different permissions, DANKE would generate different conceptual queries.

User permission configuration strategies. In what follows, assume the relational version of DANKE and recall that the tool does not support updates. Then, user permissions would apply only for SELECT operations on objects such as tables, columns, and rows. DANKE offers two strategies to configure user permissions.

In the *internal mechanisms strategy*, security handling is the responsibility of DANKE; user permissions are then defined through metadata associated with the abstract schema (see Section 4 for more details). Note that, in this strategy, DANKE is fully responsible for database security, the correct configuration of the database, and the correct use of the configuration by the DANKE Search Process.

In the *database mechanisms strategy*, security handling is the responsibility of the underlying DBMS. Thus, DANKE transmits the user identity to the DBMS, when submitting the compiled structured query for execution. This strategy relieves DANKE from the responsibility of guaranteeing that the user will never see the data that he does not have permission, but this has consequences for the implementation of access control. The *search process result security* strategy requires that user permissions be defined for the data tables and for the DANKE metadata tables and indexes. To solve this issue, the database administrator could set user permissions for metadata tables and indexes, which can be awkward. Alternatively, DANKE could generate sub-queries using the mapping of the match elements and then check the user permissions, which would affect the SEARCH module performance. By contrast, the *search process query security* strategy only requires verifying user permissions over DANKE metadata to indicate which schema elements the user has access to the data.

Both user permission configuration strategies have pros and cons, and the best alternative will depend on: the specific domain; the original security configuration of the database; and what access control implementation strategy is adopted.

4. Implementation

This section describes how to define the user authorities in the abstract schema and then describes some abstract security functions used in the *search process query security* and the *search process result security* strategies, as well as the implementation details of these functions for the current implementation of DANKE.

In the *internal mechanisms strategy*, the database definition setup will contain the list of possible authority values (resources in RBAC) for the database.

```
<setup>
    ...
    <authorities>
        <authority>auth 1</authority>
        <authority>auth 2</authority>
    </authorities>
</setup>
```

To define the entities and properties of the abstract schema, we have to add the authorities that granted access to them.

```
<property id="property1" ...>
    ...
    <authorities>
        <authority>auth 1</authority>
        <authority>auth 2</authority>
    </authorities>
</property>
```

To define authorities at the row level, we need to add an attribute in the tables with the values of the authorities for each row. In the ABSTRACT SCHEMA, we need to create a property with a specific attribute `type="authorities"` that will be mapped to an attribute in the table.

```
<create_properties>
  ...
  <property id="entity1.authorities" type="authorities" ... />
  ...
</create_properties>

<map_properties>
  ...
  <map property="entity1.authorities">
    <tables>table1</tables>
    <inlineFunction>table1.attribute1</inlineFunction>
  </map>
  ...
</map_properties>
```

The creating/updating process of DANKE database is responsible for reading the definition and creating the metadata tables to represent the ABSTRACT SCHEMA (tables with information of entities, properties, joins, and the physical map to the database of these elements). It is also responsible for indexing, transforming, and enriching the original data.

The structure of tables in DANKE database will be modified to add the `authorities` values. The main changes are:

- Add the column `authorities` to the schema metadata tables for entity and property.
- Add the column `authorities` to all data tables.

The creating/updating process of DANKE database will handle the changes as described:

- It will read the tag `authorities` from the entities and properties definition and set the values in the column `authorities` of the metadata tables for entity and property.
- It will also add the properties with `type="authorities"` property metadata table along with the property map definition.

As DANKE is a platform that can be instantiated for any domain, the architecture would be flexible and guarantee security in the search process for the *search process result security* strategy and the *search process query security* strategy, using security configuration by database mechanisms, or security configuration by the internal mechanisms of DANKE itself.

Thus, we define abstract functions that will handle security. These functions will be parameters for the search process in the *search process result security* strategy and in the *search process query security* strategy, and can be defined based on the security configuration adopted in the specific domain:

- The function *Filter Security* checks if a user has access to a specific schema element: `Filter Security:(schema element, user) → boolean`.
- The function *Security Connection* returns a connection to grant the security of the data to be recovered: `Security Connection:(user) → connection`.
- The function *Granted Security* modifies the conceptual query, if necessary, to grant the security of the data to be recovered.: `Granted Security:(conceptual query, user) → (conceptual query)`.

For the *database mechanisms strategy*, we define the functions as:

- *Filter Security*: queries the database, creating a structured query with the materialization of the property or entity, to check if the user has access to the element.
- *Security Connection*: creates a new connection with the user identity.
- *Granted Security*: does not modify the conceptual query, it is the same as the input.

For the *internal mechanisms strategy*, we define the functions as:

- *Filter Security*: uses the schema to recover the *authorities* of the element and compares with the user's *authorities*.
- *Security Connection*: creates a default connection with access to the whole database.
- *Granted Security*: modifies the conceptual query, if necessary, to guarantee security. To obtain the result conceptual query:
  - Using the function *Filter Security*, the entities and properties that the user has no access going to be removed. This guarantees tables and attributes security.
  - Filters, with the values of the user's authorities, over the properties with `type="authorities"` associated with entities in the original conceptual query will be added. This guarantees the security of the rows.

Due to the flexibility of the proposed architecture, we can create new definitions adapted for other specific scenarios.

Figure 2 shows the search process for the *search process result security* strategy. The SEARCH module does not require any security check. In the QUERY-PROCESSING module, the step that compiles the conceptual query using the function *Granted Security* could modify the query to guarantee security, and the step that executes the structured query uses the function *Security Connection* to create the connection to the database management system according to the security configuration.

To facilitate the integration of the proposed platform with different systems, for the *search process result security* strategy the values of the user permissions are passed as parameters. The security abstract functions should be implemented to handle these permissions. In our implementations, we use a Certificate Authority (CA) server, where the user and his permissions (roles and resources) are defined. To submit any search, the user must be logged on so that, using the CA server, we can recover his permissions and pass them to the *search process result security* strategy.

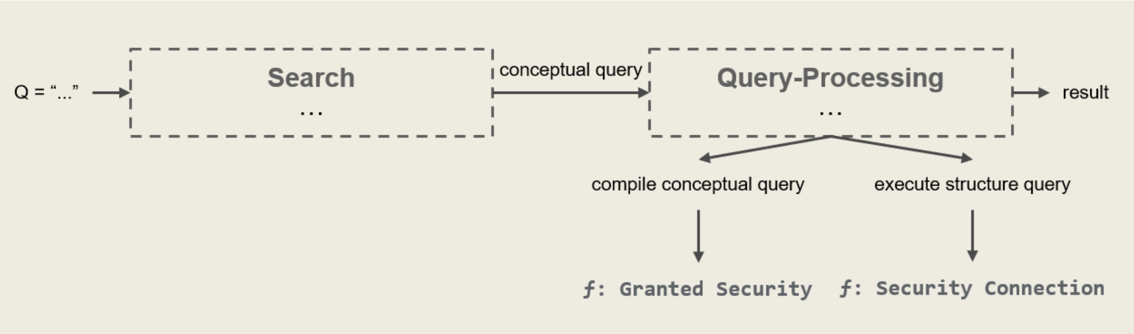


Figure 2. Search Process Result Security

Figure 3 shows the search process for the *search process query security* strategy. In the SEARCH module, the step that finds the matches with the keywords uses the functions *Filter Security* and *Security Connection* to recover only matches with elements and values that the user has permission to access, which guarantees that the conceptual query generated is going to be viable for the user. In the QUERY-PROCESSING module, the step that executes the structured query uses the function *Security Connection* to create the connection to the database management system according to the security configuration.

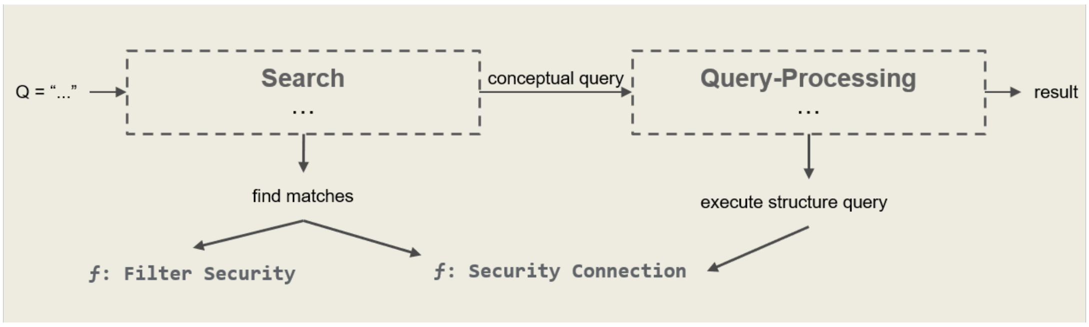


Figure 3. Search Process Query Security

5. Use Cases

This section details two use cases using a public database and an industrial database, both relational databases, that incorporate security configurations via the DANKE internal mechanisms.

5.1. COVID-SUS

Figure 4 shows a mind map snippet of the COVID-SUS dataset, where solid blank-filled circles represent entities, solid colored-filled circles represent properties, and rectangles indicate some existing data values.

According to the schema entities, we define the following resources:

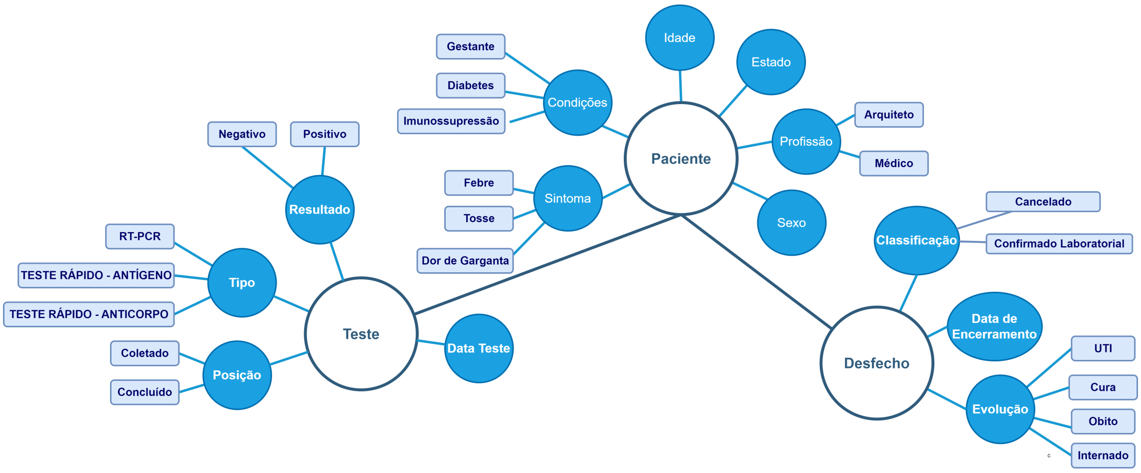


Figure 4. A mind map snippet of the COVID-SUS dataset.

- `schema`: grants access to the whole dataset.
- `teste`: grants access to the whole entity *Teste*.
- `desfecho`: grants access to the whole entity *Desfecho*.
- `paciente-rj`: grants access to data of the entity *Paciente* with rows associated with the state *Rio de Janeiro*. It does not grant access to the property *Profissão*.

For the current use case, we define three roles. The role *funcionario*, for users with the maximum privileges. The role *doutor-rj*, for doctors that work in the state *Rio de Janeiro*. Finally, the role *analisador* for users responsible for laboratory exams. The mapping of roles with resources is configured as shown in Table 1.

Roles/Resources	`schema`	`teste`	`desfecho`	`paciente-rj`
`funcionario`	x			
`doutor-rj`		x	x	x
`analisador`		x		

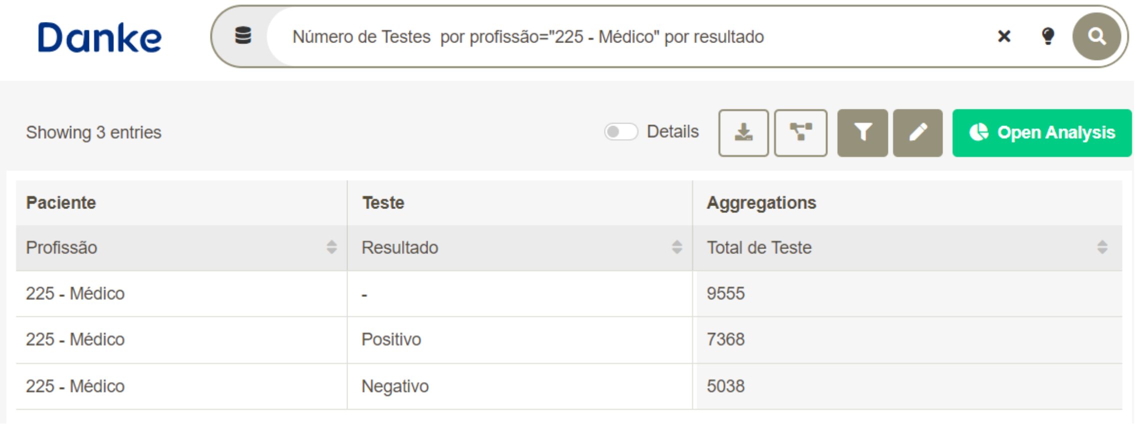
Table 1. The mappings of roles with resources for the COVID-SUS Database.

We present a list of queries and the different results based on user roles.

QUERY 1. **Número de Testes por profissão="225 - Médico" por resultado**

Figure 5 shows the result for the user with role *funcionario*, and Listing 1 details the SQL query. The users with roles *doutor-rj* and *analisador* receive the same table without the property *Profissão*. The resource *paciente-rj* of users with role *doutor-rj* does not allow access to the property *Profissão*. The users with role *analisador* do not have access to any data from the entity *Paciente*. Note that the aggregation calculation is also affected because the data of *Profissão* are fully removed from the final SQL query executed.

Listings 2 and 3 detail the SQL queries for the users *doutor-rj* and *analisador*, respectively. These queries do not have the clause *COVID_SUS_PACIENTE.CBO* in neither of the two SELECT and the GROUP BY, because there are attributes associated with property *Profissão*.



Showing 3 entries

Paciente	Teste	Aggregations
Profissão	Resultado	Total de Teste
225 - Médico	-	9555
225 - Médico	Positivo	7368
225 - Médico	Negativo	5038

Figure 5. Result for the query: `Número de Testes por profissão="225 - Médico" por resultado`

Listing 1. QUERY 1 for funcionario

```
SELECT PACIENTE.CBO, TEST.RESULTADOTESTE,
  COUNT(TEST.META_REPCOL)
FROM ( SELECT PACIENTE.META_REPCOL, PACIENTE.CBO, TEST.META_REPCOL, TEST.RESULTADOTESTE
  FROM TEST, PACIENTE
  WHERE (PACIENTE.GID = TEST.GID AND PACIENTE.CBO = '225-Medico'
    AND (TEST.AUTHORITIES LIKE '%schema%' OR TEST.AUTHORITIES IS NULL)
    AND (PACIENTE.AUTHORITIES LIKE '%schema%' OR PACIENTE.AUTHORITIES IS NULL)))
GROUP BY PACIENTE.CBO, TEST.RESULTADOTESTE
```

Listing 2. QUERY 1 for doutor-rj

```
SELECT TEST.RESULTADOTESTE, COUNT(TEST.META_REPCOL)
FROM ( SELECT PACIENTE.META_REPCOL, TEST.META_REPCOL, TEST.RESULTADOTESTE
  FROM TEST, PACIENTE
  WHERE (PACIENTE.GID = TEST.GID  AND PACIENTE.CBO = '225-Medico'
    AND (TEST.AUTHORITIES LIKE '%teste%' OR TEST.AUTHORITIES LIKE '%desfecho%'
      OR TEST.AUTHORITIES LIKE '%paciente-rj%' OR TEST.AUTHORITIES IS NULL)
    AND (PACIENTE.AUTHORITIES LIKE '%teste%' OR TEST.AUTHORITIES LIKE '%desfecho%'
      OR PACIENTE.AUTHORITIES LIKE '%paciente-rj%' OR PACIENTE.AUTHORITIES IS NULL)))
GROUP BY TEST.RESULTADOTESTE
```

Listing 3. QUERY 1 for analisador

```
SELECT TEST.RESULTADOTESTE, COUNT(TEST.META_REPCOL)
FROM (SELECT PACIENTE.META_REPCOL, TEST.META_REPCOL, TEST.RESULTADOTESTE
  FROM TEST, PACIENTE
  WHERE (PACIENTE.GID = TEST.GID  AND PACIENTE.CBO = '225-Medico'
    AND (TEST.AUTHORITIES LIKE '%teste%'  OR TEST.AUTHORITIES IS NULL)
    AND (PACIENTE.AUTHORITIES LIKE '%teste%' OR PACIENTE.AUTHORITIES IS NULL)))
GROUP BY TEST.RESULTADOTESTE
```

5.2. An Industrial Dataset

As a second use case, we use an industrial dataset. Figure 6 shows a snippet mind map of the dataset, where solid blank circles represent entities, solid filled circles represent properties, and rectangles indicate some existing data values.

According to the schema entities, we define the following resources:

- `schema`: grants access to the whole dataset.

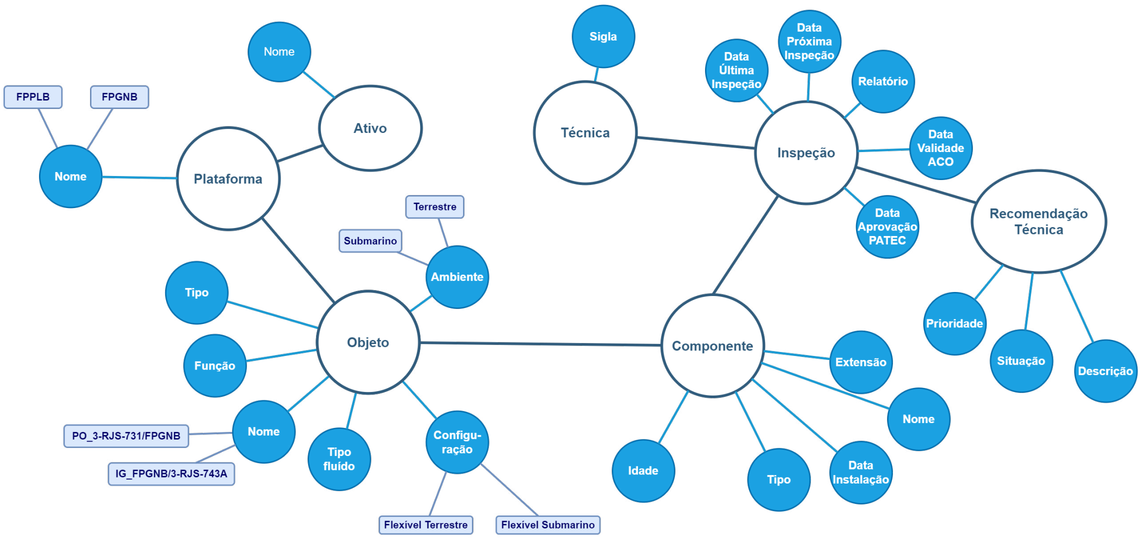


Figure 6. The snippet mind map of the industrial dataset.

- `entities`: grants access to the the entities *Ativo*, *Técnica*, *Componente*, *Recomendação Técnica*
- `plat-fpplb`: grants access to data of the *Plataforma* FPPLB (only this row of the entity *Plataforma*).
- `insp`: grants access to data of the entity *Inspeção*.
- `obj-simple`: grants access to the data of the entity *Objeto* except for the data of properties *Ambiente* and *Configuração*.

For the use case, we define three roles. The role *admin*, for users with the maximum privileges. The role *user-fpplb*, for users that works in the *Plataforma* FPPLB. Finally, the role *user-insp* for users responsible for inspection. Table 2 shows the mappings of roles with resources.

Roles/Resources	`schema`	`entities`	`plat-fpplb`	`insp`	`obj-simple`
`admin`	x				
`user-fpplb`		x	x	x	x
`user-insp`				x	x

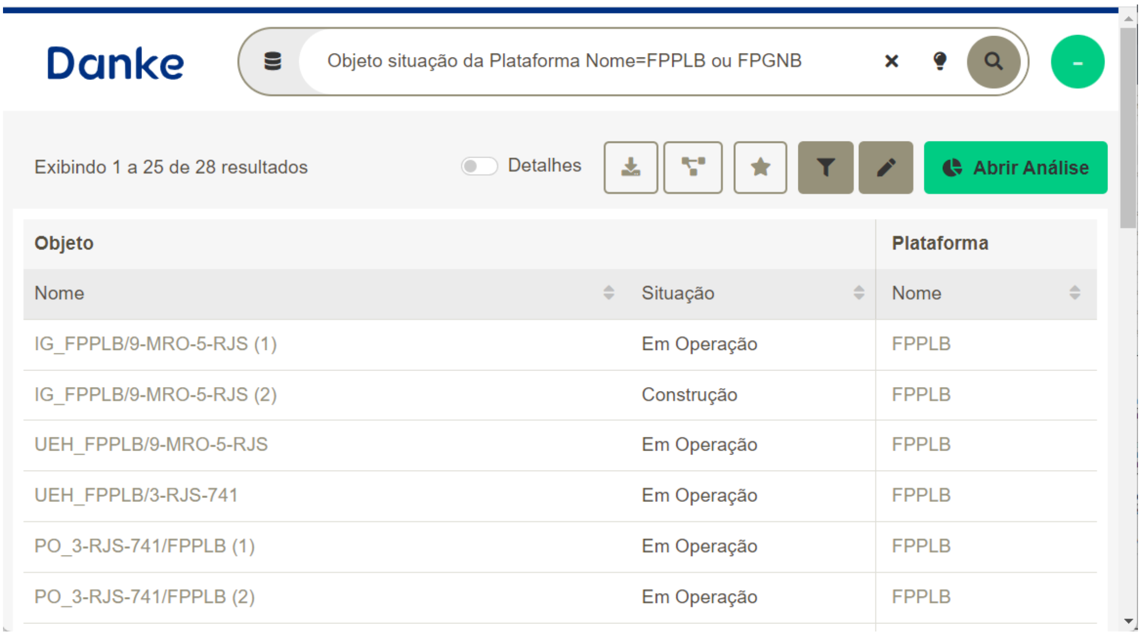
Table 2. The mappings of roles with resources for the Industrial Database.

We next present an example of the different results based on user roles.

QUERY 2: **Objeto situação da Plataforma Nome = FPPLB ou FPGNB**

Figure 7 shows the result for the user with role *admin*, and Listing 4 details the SQL query. The user with role *user-fpplb* receives the same table result but with fewer lines because s/he only has access to the rows with value *fpplb* for *Plataforma*. Listing 5 details the SQL query. This query is similar to the SQL query in Listing 4 due to the user permissions, except for the security check for row level.

The user with role *user-insp* receives the table without the property *Nome* of entity *Plataforma* because the resources *insp* and *obj-simple* do not allow access to



Objeto situação da Plataforma Nome=FPPLB ou FPGNB

Exibindo 1 a 25 de 28 resultados

Detalhes

Objeto		Plataforma
Nome	Situação	Nome
IG_FPPLB/9-MRO-5-RJS (1)	Em Operação	FPPLB
IG_FPPLB/9-MRO-5-RJS (2)	Construção	FPPLB
UEH_FPPLB/9-MRO-5-RJS	Em Operação	FPPLB
UEH_FPPLB/3-RJS-741	Em Operação	FPPLB
PO_3-RJS-741/FPPLB (1)	Em Operação	FPPLB
PO_3-RJS-741/FPPLB (2)	Em Operação	FPPLB

Figure 7. Result for the query: `Objeto situação da Plataforma Nome=FPPLB ou FPGNB`

that data. Listing 6 details the SQL query. This query does not have the clause `PLATAFORMA.META_REPCOL` in the SELECT clause since it is an attribute associated with the entity *Plataforma*.

Listing 4. QUERY 2 for admin

```
SELECT OBJETO.META_REPCOL, OBJETO.ISTA_TX_SIT_DUTO, PLATAFORMA.META_REPCOL
FROM PLATAFORMA, OBJETO
WHERE (PLATAFORMA.NOME = OBJETO.ISTA_NM_INSTALACAO_RESP)
  AND ((PLATAFORMA.META_REPCOL = 'fpplb') OR (PLATAFORMA.META_REPCOL = 'fpgnb'))
  AND ((OBJETO.AUTHORITIES LIKE '%schema%') OR (OBJETO.AUTHORITIES IS NULL))
  AND ((PLATAFORMA.AUTHORITIES LIKE '%schema%') OR (PLATAFORMA.AUTHORITIES IS NULL))
```

Listing 5. QUERY 2 for user-fpplb

```
SELECT OBJETO.META_REPCO, OBJETO.ISTA_TX_SIT_DUTO, PLATAFORMA.META_REPCOL
FROM PLATAFORMA, OBJETO
WHERE (PLATAFORMA.NOME = OBJETO.ISTA_NM_INSTALACAO_RESP)
  AND ((PLATAFORMA.META_REPCOL = 'fpplb')  OR (PLATAFORMA.META_REPCOL = 'fpgnb'))
  AND ((OBJETO.AUTHORITIES LIKE '%entities%') OR (OBJETO.AUTHORITIES LIKE '%plat-fpplb%')
    OR (OBJETO.AUTHORITIES LIKE '%insp%')
    OR (OBJETO.AUTHORITIES LIKE '%obj-simple%') OR (OBJETO.AUTHORITIES IS NULL))
  AND ((PLATAFORMA.AUTHORITIES LIKE '%entities%')
    OR (PLATAFORMA.AUTHORITIES LIKE '%plat-fpplb%')
    OR (PLATAFORMA.AUTHORITIES LIKE '%insp%')
    OR (PLATAFORMA.AUTHORITIES LIKE '%obj-simple%')
    OR (PLATAFORMA.AUTHORITIES IS NULL))
```

Listing 6. QUERY 2 for user-insp

```
SELECT OBJETO.META_REPCO, OBJETO.ISTA_TX_SIT_DUTO
FROM PLATAFORMA, OBJETO
WHERE (PLATAFORMA.NOME = OBJETO.ISTA_NM_INSTALACAO_RESP)
  AND ((PLATAFORMA.META_REPCOL = 'fpplb') OR (PLATAFORMA.META_REPCOL = 'fpgnb'))
  AND ((OBJETO.AUTHORITIES LIKE '%insp%')
    OR (OBJETO.AUTHORITIES LIKE '%obj-simple%') OR (OBJETO.AUTHORITIES IS NULL))
  AND ((PLATAFORMA.AUTHORITIES LIKE '%insp%')
    OR (PLATAFORMA.AUTHORITIES LIKE '%obj-simple%') OR (PLATAFORMA.AUTHORITIES IS NULL))
```

6. Conclusions and Directions for Future Work

This paper addressed the access control problem in the context of database keyword search. It detailed the solutions implemented for DANKE, a platform that offers keyword search over relational databases and RDF datasets. The solutions explored the use of database mechanisms as well as internal mechanisms of the platform. Basically, the solutions either generate conceptual queries based on user permissions, or filter query results based on user permissions, ensuring that users can only access authorized data. Finally, the paper detailed two use cases to validate the proposed solution, including an industrial application.

Acknowledgements

This work was partly funded by FAPERJ under grants E-26/200.834/2021, by CAPES under grant 88881.134081/2016-01 and 88882.164913/2010-01, by CNPq under grant 305.587/2021-8 and by Libra Consortium (Petrobras, Shell Brasil, Total Energies, CNOOC, CNPC, and PPSA) within the ANP R&D levy as a commitment to research and development investments.

References

Bertino, E., Bonatti, P. A., and Ferrari, E. (2000). Trbac: a temporal role-based access control model. *RBAC '00: Proceedings of the fifth ACM workshop on Role-based access control*, pages 21–30.

Bertino, E., Ghinita, G., and Kamra, A. (2011). Access control for databases: Concepts and systems. *Foundations and Trends® in Databases*, 3:1–148.

Bertino, E. and Sandhu, R. (2005). Database security-concepts, approaches, and challenges. *IEEE Transactions on Dependable and Secure Computing*, 2:2–18.

Damiani, M. L., Bertino, E., Catania, B., and Perlasca, P. (2007). Geo-rbac: A spatially aware rbac. *ACM Transactions on Information and System Security (TISSEC)*, 10.

Ferraiolo, D. F., Kuhn, D. R., and Chandramouli, R. (2001). Proposed nist standard for role-based access control. *ACM Transactions on Information and System Security*, 4:224–274.

García, G. M. (2020). *A Keyword-based Query Processing Method for Datasets with Schemas*. PhD thesis, Thesis presented to the Graduate Program in Informatics, PUC-Rio (March 2020 ....

Izquierdo, Y. T., Garcia, G. M., Lemos, M., Novello, A., Novelli, B., Damasceno, C., Leme, L. A. P. P., and Casanova, M. A. (2021). A platform for keyword search and its application for covid-19 pandemic data. *Journal of Information and Data Management*, 12(5).

Izquierdo, Y. T., García, G. M., Lemos, M., Novello, A. F., Novelli, B., Damasceno, C., Leme, L. A. P. P., and Casanova, M. A. (2020). Keyword search over the covid-19 data. In *SBBD*, pages 205–210.

Jabal, A. A., Davari, M., Bertino, E., Makaya, C., Calo, S., Verma, D., Russo, A., and Williams, C. (2019). Methods and tools for policy analysis. *ACM Comput. Surv*, 51.

Kumar, M. and Newman, R. E. (2006). Strbac - an approach towards spatio-temporal role-based access control. *Communication, Network, and Information Security*, 155.

Sandhu, R., Ferraiolo, D., and Kuhn, R. (2000). The nist model for role-based access control: Towards a unified standard. *ACM workshop on Role-based access control*, 10.

Sandhu, R. S., Coyne, E. J., Feinstein, H. L., and Youman, C. E. (1996). Computer role-based access control models. *Computer*, 29:38–47.