

Implementation of the Concept of a Repository for Automated Processing of Semi-Structural Data

Mateusz Piech, Bartosz Rakoczy, Jacek Dajda, and Marek Kisiel-Dorohinicki

AGH University of Science and Technology, Cracow, Poland

<https://doi.org/10.26636/jtit.2020.136919>

Abstract—Semi-structural data tend to be problematic due to the sparsity of their attributes and due to the fact that, regardless of their type, they are immensely diverse. This means that data storage is a challenge, especially when the data contained within a relational database – often a strict requirement defined in advance. In this paper, we present a thoroughly described concept of a repository that is capable of storing and processing semi-structural data. Based on this concept, we establish a database model comprising the architecture and the tools needed to search the data and build relevant processors. The processor described may assign roles and dispatch tasks between the users. We demonstrate how the capacities of this repository are capable of overcoming current limitations by creating a system for facilitated digitization of scientific resources. In addition, we show that the repository in question is suitable for general use, and, as such, may be adapted to any domains in which semi-structural data are processed, without any additional work required.

Keywords—document management system, ECM, JSON, workflow.

1. Introduction

Formulation of the concept of a repository for processing semi-structural data is extremely important when attempting to process data with an unknown structure, at early stages of the system's implementation process. Currently, no SQL database engine offers such a functionality, since the given requirements related the system rule out such a possibility. Data stored in a repository have a complex hierarchical model which varies between the specific data sets, regardless of their type, and thus leads to the sparsity of data. It may also be classified as semi-structural, meaning that it is characterized by a tree-like structure.

Our main motivation for establishing this concept was to create a repository allowing to process hierarchical data within a relational database. In addition to ensuring efficient storage and management of data, an effective repository must contain a set of features that have never been seen before in similar systems, and must serve as a framework for advanced automation of processing. In our concept, many technical issues needed to be resolved, many with no

obvious or previously devised solutions. These include the following:

- creation of an adequate database enabling storage and procurement of semi-structural. Additionally, the system needed to offer a functionality allowing to add novel types of resources online, regardless of their structure;
- creation of a query language, enabling to look up the resources;
- creation of a framework for assembling and managing the resource processing pipeline;
- establishment of a hierarchical data access strategy providing access to individual tasks in the processing pipeline.

The literature presents systems that have so far been used mostly for storing semi-structural data. In contrast, in systems meant for automated processing, i.e. in enterprise content management (ECM) solutions, structural data are used. In our concept, we offer an innovative mix of both of these types of systems, taking a novel approach to ECM tools.

From our research, there emerged the concept of a system that solves all aforementioned problems. In order to validate this conceptual solution, we implemented it by building a system used to store and process digital scientific resources. We formulated specific processing rules and a repository structure which is universal and may be applied, without any restrictions, to any domains relying on semi-structural data.

In this article, the most important elements of this system are presented, addressing the issues referred to above. Initially, we present previous work that has served as an inspiration for the creation of specific system modules. Next, we present the terminology required to understand the concept repository model and its additional elements, such as the database model and system architecture. Following that, two important aspects are explained: the query language for semi-structural sparse data and the creation of a processing pipeline. Lastly, the concept of the system is evaluated and summarized, and the existing solutions and specific ideas for future improvements are discussed.

2. Related Work

One of the first systems for semi-structural data management was Lore [1]. It was designed for storing objects with a tree structure. In their work, the authors thoroughly analyzed the system: starting with the user interface, to indexing, querying and query optimization, to physical data storage in the database. The next iteration of the system, as shown in [2], migrated the model along with the corresponding query language to an XML-based one. This resulted in simplification of the system and transferring the responsibility for keeping the model to the database engine. Research concerned with using native XML in relational databases, initiated then in [3], resulted in its eventual commercial implementation, leading to the first ever native solution – XML Support in Microsoft SQL Server 2005 [4]. Subsequent research on the use of XML in relational databases resulted in the development of a query language, SQLxD [5], which aimed to execute searches through XML documents in a transparent way, without using specific XML operators.

With the rapid expansion of amounts of data transferred via the Internet, the XML format was displaced by JSON whose performance and compactness were superior compared to XML [6], [7]. Those features encouraged its implementation in terms of native support in relational databases, starting with PostgreSQL 9.2 in 2012. When tested semi-structural data scenarios in [8], it outperformed the solution based on the open data scheme (entity-attribute-value model), attaining results similar to those of a document database (MongoDB) with regards to performance.

The interest in creating a repository for semi-structural data storage, and in processing this type of data, drove other research in the field. One of the most intriguing papers on this issue is [9], in which storage for semi-structural data was created, and in which the data were connected by a graph, with the intention of facilitating data lookup for data mining purposes. Another repository, designed as a digital library, was project Aquarelle [10], in which SGML structure objects were processed into schemes defined by object-oriented semantic network systems. Other than simple repositories for semi-structural data, there are also some general systems capable of working with this sort of data, such as: Sineu [11] – SQL system for multi-structured data, BioRegistry [12] – structured metadata repository for bioinformatic databases, or a scalable analysis platform for semi-structured data [13].

Some analyses focused also on methods for semi-structural data storage [14]–[16], as well as on approaches to unstructured data access and information extraction (from HTML [17] or XML documents [18], [19]). Other researchers created a new model, ORA-SS [20], to which XML data were parsed. There was also an attempt to create a layered view model for XML Repositories [21], and a wrapper for XML [22]. Various ways of processing queries and of optimizing XML repositories were also explored in [23], [24], in addition to the concept of processing

data via the ETL process [25]. Despite the abundance of studies on semi-structural data processing, there is still an apparent lack of research focusing on our topic of interest, i.e. on both semi-structural data storage in JSON format in a repository, and suitable technologies enabling the repository to perform automated processing.

3. Related Systems

The concept we have developed may be categorized as being of the ECM class, meaning that it combines systems with solutions for:

- document management (DMS),
- electronic document flow (Workflow),
- business process management (BPM).

Nowadays, numerous systems similar to the one presented here exist. One of the most important BPM systems on the market is Metasonic Flow [26], which assumes that the main units in the system are tasks organized into processes. It is possible to assign tasks to users or to set tasks that are to be automatically performed by the system (e.g. report generation).

Doxis4 iECM is another noteworthy example [27], with its core underlying idea consisting in offering various database services (e.g. authorization, logs, data management), i.e. elements that contribute to the overall end solution. One of the advantages of this system is its ability to search data based on metadata filters.

Both systems mentioned above, as well as other concerned with this specific area, share a number of peculiar characteristics (i.e. structuring tasks into processes and assigning these to users), while only differing in terms of their features. Conclusions reached based on the analysis of these systems served, to a certain degree, as a point of departure for our own work, as we noticed that none of these systems supported the processing of dynamic structure data.

4. Architecture and Database Model

In the first stage of this research, we focused on technical aspects [28], which required the following tasks to be performed:

- the formulation of a definition of specific elements used in the system,
- preparation of a database model,
- preparation of a system architecture.

4.1. Terms and Definitions

The analysis of the problem revealed that a semi-structured repository contained two kinds of elements: resources (physical objects) and relationships (relations between ob-

jects, i.e. resources). Each element has its own template which defines its type, along with its structure, consisting of a metadata definition that contains information about restrictions (constraints, regexp, etc.), along with its position (hierarchy) within the data tree. The metadata definition maintains the schema and permits the control of metadata values which are based on both resources and relationships.

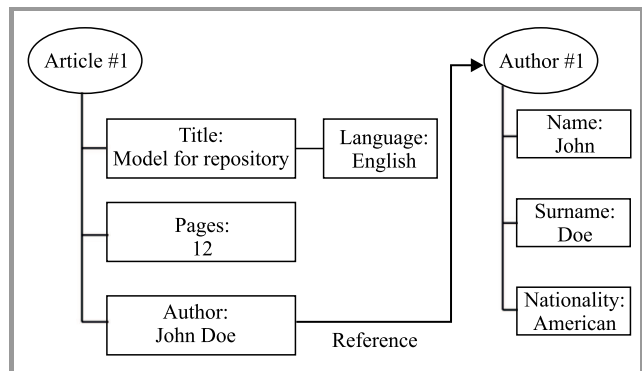


Fig. 1. Schema depicting two dynamic objects and their relationship within the repository.

The relationships mentioned above allow for the connection of objects at two levels, making it possible to join the individual types in the following ways:

- resource-resource: a simple connection of two resources, for example Article#1 is connected with Article#1 because of a quotation,
- metadata-resource: a connection between a metadata value and another resource it points to.

A simple example of a metadata-resource connection is shown in Fig. 1, based on the data from a digital resource repository.

4.2. Database Model

The next step for us was to create a schema model to work with the previously described objects, and to design the presented solution, in compliance with the applicable technological requirements, in a relational database. Otherwise, we would have chosen a document-oriented database – a member of the NoSQL family [29], designed for storing semi-structural objects. Relational databases have two ways of storing such data, namely an open schema model or a dedicated, native model (e.g. XML or JSON). The entity-attribute-value model [30], which stores objects in three tables, as per the model’s name, is a practical example of the former variety. Another example would be the inverted index model [31], which, despite being intended for storing structure mapping as an index, contains a model allowing for semi-structural data storage, thanks to its ability to forge references between objects and attributes. For the purpose of this particular concept, based on the

satisfactory results obtained in the course of research conducted [6], [8], [32], [33], we chose a dedicated, native method, namely JSON.

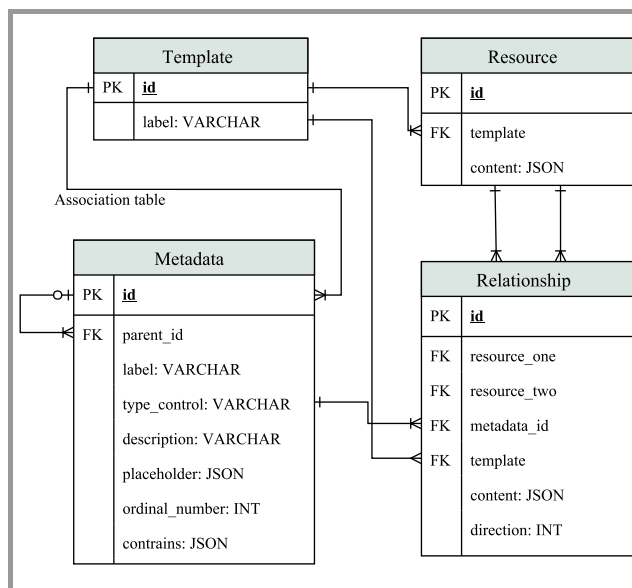


Fig. 2. Model schema for a dynamic repository in a relational database.

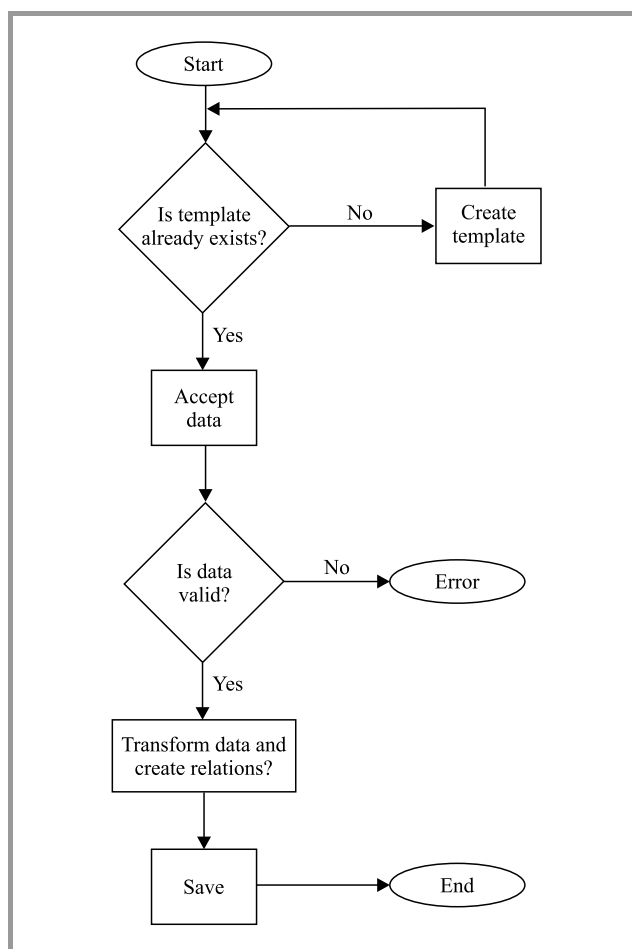


Fig. 3. Flow of data being pushed into the repository.

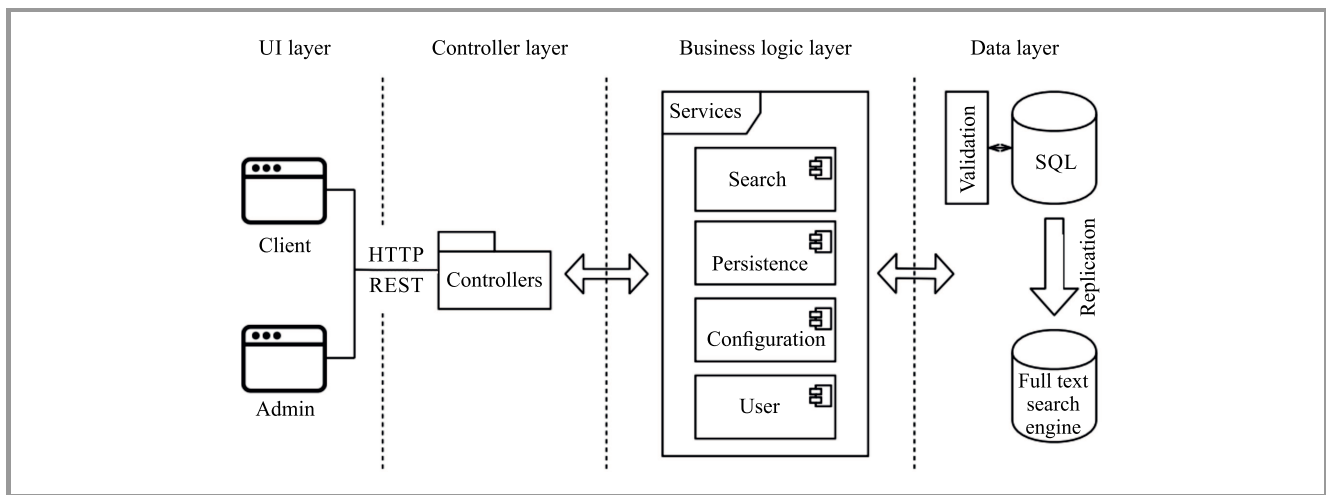


Fig. 4. Final architecture of the repository.

In our initial schema, each element (resource, relationship, template and metadata) had its own table, as shown in Fig. 2. Unfortunately, the schema itself failed to meet all of the authors' expectations, since it lacked the ability to control the structure of any data uploaded. To solve this issue, we created an additional layer with a validation mechanism and object transformation abilities deployed during all uploads. The flow of data is shown in Fig. 3, and its in-depth description may be found in [28]. It is run both for resources and for relationships.

4.3. Repository Architecture

The concept of the system architecture was based on layers which separate specific responsibilities. Four main layers may be distinguished:

UI layer – this repository access layer was split into two independent applications. The first one, called the client, allows to explore the repository content. The other, called administrative, has much more critical responsibilities. Depending on their rights, the user is granted access to their tasks in the repository, as created during data stream processing.

Controller layer – defines the communication between UI and the repository. The communication itself is based upon REST, with an amendable API to enable future improvements for the purpose of other types of UI applications, and for various platforms.

Business logic layer – provides the repository's main logic. Its key components are:

- search service – for processing data queries. Here, an implementation of the query language (described in Section 5) may be found;
- persistence service – a component for processing data uploaded to the repository, validation level imple-

mentation and data control, as described in Subsection 4.2;

- configuration service – a module meant for the configuration of a repository specification in accordance with its purpose. This involves creating a proper template and metadata, but also data processors tied to the life cycle – in this case, the flow of documents during the digitization process;
- user service – a module for data security control and system roles.

Data layer – provides data access and comprises two elements. The first is a relational database (SQL) in which all repository data are kept. Additionally, to improve search capabilities, a full text search engine was utilized. Data are replicated between databases and their application is compliant with the CQRS design pattern [34], offering clear separation between reading and modifying the data.

5. Query Language

The primary purpose for developing a dedicated query language for this repository was to support the accessing and the filtering of resources in the database. Usually, the process of writing queries is rather complex, due to the dynamic and unknown structure of the objects. The query language adheres to the principles of data-driven development, and was developed based on the structure and the properties of data representation adopted in the presented model. To facilitate common data access scenarios, the query language works in two modes – one for syntax completion, and one for fetching resources.

The stored resources, as stated above, have a tree-like structure. Thus, it is necessary to introduce a mechanism for accessing the nested levels. While researching the already existing technologies, we encountered a template engine

known as Twig¹. One of its features offers stream-like access to the nested objects, using pipe character — which separates subsequent steps in the query. A syntax example is presented below.

```
{{TOKEN_1 | TOKEN_2 | ... | TOKEN_n}}
```

There are three types of tokens:

1. Access to resource. Possible tokens:
 - *r* – access to all resources,
 - *r(ID)* – access to one resource with a given *ID*,
 - *r(TYPE)* – access to all resources of a given *TYPE*.
2. Access to metadata. Possible tokens:
 - *m(ID)* – access to metadata with a given *ID*,
 - *m(NAME)* – access to metadata with a given *NAME*.
3. Operators. Possible tokens:
 - *first* – select the first result,
 - *sum* – summarize all results,
 - *max* – select the maximum result.

The examples of query syntax are:

- summarize the total word count at pages in a book with *id = 12*

```
{{r(12) | m('Page') | m('Words') | sum}}
```
- provide all authors of a book with *id = 12*

```
{{r(12) | m('Author') | m('Last name')}}}
```

As we can see, the query language works based on two modes, both of which are automatically translated into the auto-generated SQL language. However, the algorithm for creating Select differs, depending on the mode. In the syntax completion feature, the next nested levels are obtained by creating Join to a metadata table with the name or id condition. An example of auto-generated SQL for the first query example, after three tokens, is:

```
SELECT m2.label FROM Resource AS r
JOIN template_metadata AS tm
  ON tm.template_id = r.template
JOIN metadata AS m
  ON m.id = tm.metadata_id
JOIN metadata AS m1
  ON m1.parent = m.id
JOIN metadata AS m2
  ON m2.parent = m1.id
WHERE r.id = 12 AND m.label = 'Page'
AND m1.label = 'Words'
```

¹<https://twig.symfony.com/>

In the resource access mode, the next nested levels are obtained using the prepared nested Select along with an unwinding method, in line with the corresponding hierarchy. The SQL generated for the first query example is:

```
S1 -> SELECT jsonb_array_elements(
      content #> 'Page'
    ) AS content FROM resource WHERE id = 12;
S2 -> SELECT jsonb_array_elements(
      S1.content::jsonb #> 'Words'
    ) AS content FROM S1;
S3 -> SELECT SUM(
      S2.content::jsonb #>> 'value'
    ) FROM S2;
```

The query language presented in this section is simple and has been designed for accessing and filtering resources in the repository. Its main advantage is that it allows to create queries with no knowledge of the structure of the resource representation and the metadata hierarchy. Moreover, its extension allowing to support more operators is straightforward and simply requires the implementation of SQL generation. The drawback of this solution is related to its performance, since the auto-generated SQL queries are never as fast as those hand-written by an expert. However, due to the optimization possible via the indexes in the database, performance is fully acceptable and does not overshadow the benefits achieved as a result of the superb functionality of the query language.

6. Data Processing

The process of adding resources to the repository is pipeline-based, thereby enabling us to split it into smaller elements, and thus to distribute tasks between a larger group of people. The first step towards achieving this objective was to introduce states to the resource, an approach which provides information as to what phase of the workflow the pro-

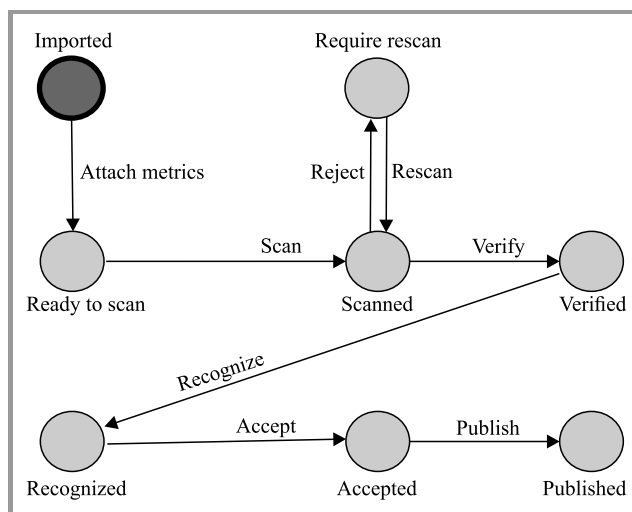


Fig. 5. The workflow of a scanned book.

cessed element finds itself in. The next step was to create a graph, showing changes between the states. An example of such a graph for the process of scanning a book in a digital archive is presented in Fig. 5. A node in such a graph represents the state which it is in. The edges contain information about the required metadata that needs to be filled in before proceeding to the next stage. Successful validation allows the state change to take place.

6.1. User Roles and Restrictions

Processing would make no sense without the ability to distribute the workload between individual system users. Bearing this in mind, the next element we adapted for the purpose of the repository was related to the roles that the users play in the system. The most important role is that of the administrator who has the ability to create new roles and assign them to users. The next role in the hierarchy is that of a coordinator who manages the process and whose task is to define roles for the respective stages, i.e. operators for process stages (e.g. scanners, graphic designers or editors). There are two restriction levels here: for the entire application or for a specific resource.

During resource processing, more restricted metadata (i.e. those that are not made public) are also filled in, for example information about the person processing the resource or internal system markings. Therefore, each process has a configured, hierarchical metadata structure which is visible in the public portion of the system.

7. Concept Application

7.1. Technical Requirements

The crucial part of the implementation of any concept involves the choice of the right relational database. Currently, all most popular relational databases support JSON:

- PostgreSQL 9.2 (2012),
- Oracle Database 12c release 1 (2013),
- MySQL 5.7 (2015),
- Microsoft SQL Server 13.00 (2016).

In the presented implementation, PostgreSQL version 10.4 was used, and our choice was influenced mostly by the comparison of the performance of the engines [36]. Even though the choice of any particular engine has no bearing on whether the presented concept is operational or not (since it only affects its performance), it was possible to use any relational database compliant with the technical requirements and in accordance with the authors' interests.

Since the requirement for a relational database was the only specific constraint we were given, the remaining parts, such

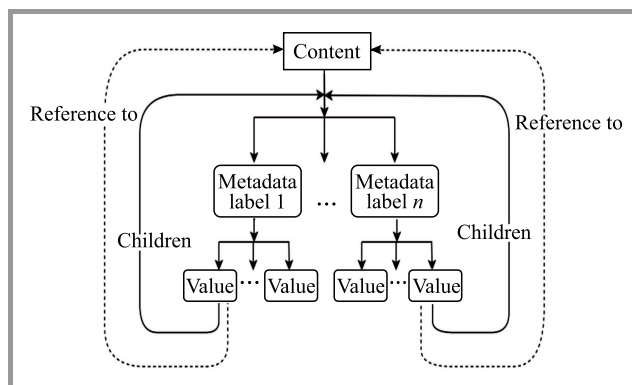


Fig. 6. A visualization of the structure used for normalization and validation of repository input data.

as UI or the business logic layer, could be chosen arbitrarily, as long as the following were implemented:

- data flow (Fig. 3),
- data structure (Fig. 6),

Such features as implementation of the query language (as described in Section 5), or use of a full text search engine were optional and affected user experience and performance only.

7.2. RePeKa

Our concept was used to implement a system for storing and processing digital science resources. Archiving science resources is a complex process, requiring numerous stages, and is dependent on the creation of a processing pipeline (workflow). It begins with the data insertion stage (scanned and processed OCR or raw data). Then, a multi-step content analysis is conducted in which different types of metadata are filled in – the authors, page count or relations to other documents. Each step is verified by a person with a proper authorization, until the resource is finally placed in the repository. The system implemented was known as RePeKa, and its user interface may be seen in Fig. 7.

7.3. Application to Science

In keeping with our motivations, we sought to implement the concept in the digitization of scientific resources. Nevertheless, this concept could be applied in multiple fields of science and areas of knowledge. One of the most relevant areas, where this work might be of benefit, is that of forensic analysis. Examples of research based on hierarchical or semi-structural data may be found in [37]–[39]. Biological sciences, for which entity-attribute-value models have been used to date, are another area where our concept may exert a major impact [41, 40]. Further potential applications are related to repositories used in of ecology [42], neuroscience [43] or telemedicine [44].

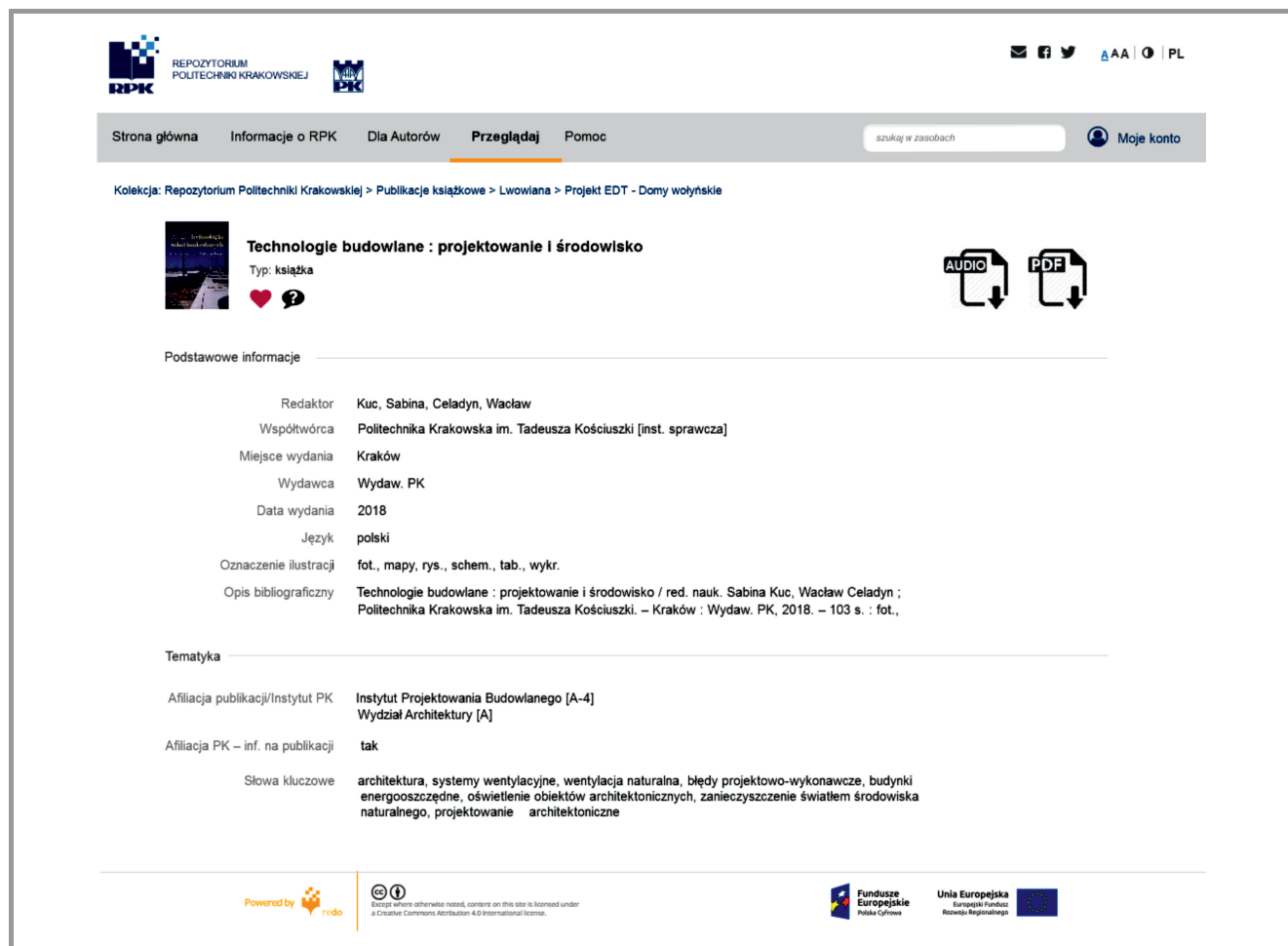


Fig. 7. RePeKa interface.

8. Evaluation

In this research, we implemented the concept in the form of the RePeKa system, and then evaluated it in terms of functionality and performance. First of all, we compared our solutions with the existing systems and, secondly, expanded the range of tests originally performed [28].

8.1. Functionality Evaluation

Comparing the RePeKa system with the solutions referred to in the related systems section, we may notice some differences in the functionalities available in each of them. Those differences are listed below:

- Data set:
 - **RePeKa** – capable of processing any data set of any structure, without requiring any changes in the system architecture. Additionally, capable of creating relations between elements of a set at any level of the structure tree;
 - **Other systems** – none of the remaining systems is capable of processing semi-structural data;

- Authorization:
 - **RePeKa** – the ability to connect any user authorization module, e.g. LDAP;
 - **Other systems** – depending on the system, internal or external modules used;
- Process automation:
 - **RePeKa** – has the capacity to connect any automatic task to the process, for example: report generation, sending an email or performing automatic optical character recognition (OCR);
 - **Other systems** – simple task automation (report generation, sending an email) available only in the Metasonic Flow system. It is, however, lacking the option to connect its own task implementation to the process, whereas RePeKa does have the capacity to do so;
- Set search-through:
 - **RePeKa** – capable of building queries of any complexity level thanks to the mechanism implemented for the query language;

- **Other systems** - metadata-based dataset searches available in the Doxis4 iECM system.

Although the concept has been implemented in the form of the RePeKa system which was less advanced when compared with Metasonic Flow or *Doxis4 iECM*, it offered the most important functionalities required for completing the task identified - i.e. ensuring digital document workflow. This is due to the capabilities of both systems which (when compared with each other) render RePeKa superior in terms of the features available. These features, coupled with the option of keeping datasets of a semi-structural nature, mean that our system is a noteworthy candidate when choosing a repository for the processing of this type of data.

8.2. Performance Evaluation

The database model presented here has been tested in previous research [28]. It was compared with the entity-attribute-value model in terms of memory consumption and performance. Both models were implemented in the same database (PostgreSQL), and both approaches offered the same functionalities. The solutions were tested for many different data sets of various sizes structure complexities. Three test cases were also prepared to simulate real case scenarios for the repository. A summary of the results of tests performed with the use of the same hardware configuration are as follows:

- on average, a twofold improvement in terms of data writing and disk size requirements was attained,
- an increase in speed of 2 to 4 times was achieved, depending on the query complexity, with better performance reached for queries including object relations.

Additionally, along with the introduction of new functions, such as the query language (Section 5), we tested the performance of the system for different data models: the one presented in this concept, the one based on entity-attribute-value, and the MongoDB-based solution (which is a document database). Tests were performed on the same hardware configuration as that mentioned previously (3.2 GHz quad-core Intel i5-4460 processor with 16 GB of RAM and 256 GB of Intel solid-state storage), on a data set consisting of 10 million objects and 10 million relations between them. We chose two test cases for the query language: one for syntax completion and the other for content fetching.

8.3. Syntax Completion

The following queries were used to test syntax completion performance:

- syntax completion for all objects of a given type (QT1):

$$\{\{r('Book') \mid m('Page') \mid m('Words') \mid * \}\}$$

- syntax completion for a single object (QT2):

$$\{\{r(12) \mid m('Page') \mid m('Words') \mid * \}\}$$

The time required to reach the next stage (separated by the pipe character “|”) was measured. There are 3 stages in each query, marked S_1 , S_2 and S_3 . The results are displayed in Figs. 8 and 9. One may notice that for QT2, the results assume similar values - with our model having the

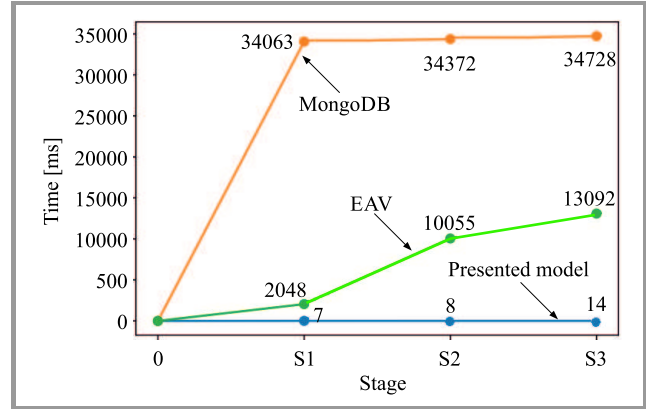


Fig. 8. Results of the QT1 tests.

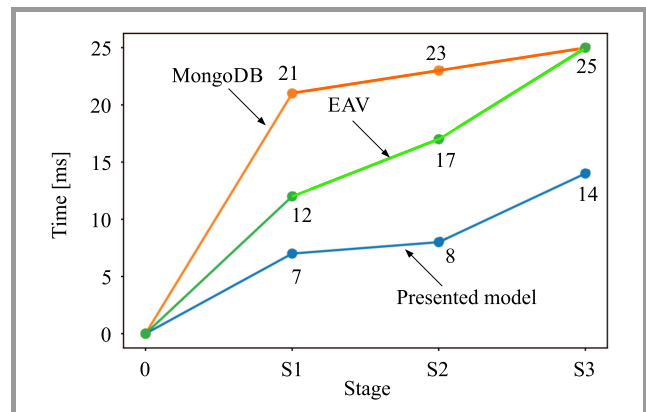


Fig. 9. Results of the QT2 tests.

advantage. In QT1, however, this concept (containing information about the potential structure) highly outperforms the others. For the proposed model, the size of the dataset does not impact the time required for processing. In other models, the effect of dataset size is more apparent - in the case of EAV, the time linearly dependent on query complexity, while for MongoDB, this process required a lookup of the entire set, and was the major cost of the whole operation (processes from 0 to S_1).

8.4. Content Fetching

In QT1 and QT2 queries, we exchanged the “*” sign for aggregate sum operand and ran the tests, marking them Q1 and Q2. The results, presented in Figs. 10 and 11, show that MongoDB yields the best results in the case of query Q2. This was to be expected, since it is a document-oriented

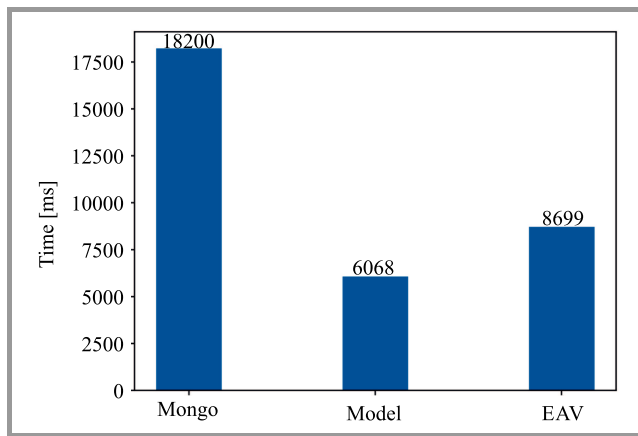


Fig. 10. Results of the Q1 tests.

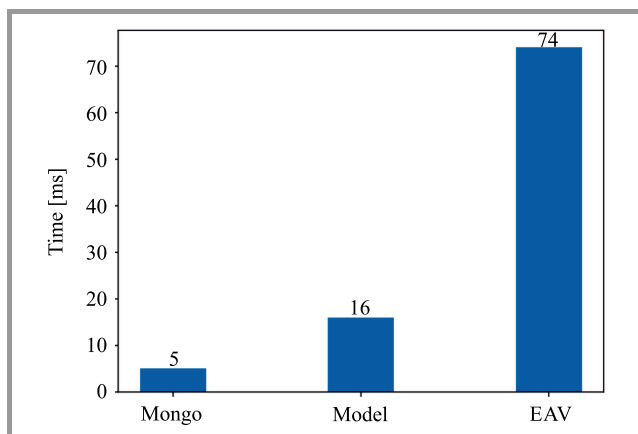


Fig. 11. Results of the Q2 tests.

database. However, Q1 relational databases perform better (due to the huge number of rows to aggregate), and the pro-posed model, using JSON, is faster than EAV (which is also true for Q2).

9. Conclusion

The objective of this research was to formulate a repository concept used for processing resources based on semi-structural data. The main achievements of our repository concept are presented below:

- a semi-structural database model, allowing for the storage of resources with a hierarchical structure. Furthermore, the open schema model facilitates the configuration of any resource type – thanks to this feature, the resulting solution is universal and thus its implementation is feasible in any domain;
- the query language for resource exploration. Its creation was made possible via the introduction of structure control for the respective resources kept by the database;
- its advanced data processor configurator, based on state changes, coupled with the configuration of roles

responsible for the respective stages. In concert with the remaining features of the concept, this makes the design of any data flow configuration possible.

9.1. Future Work

Implementation of this novel system has proven that the goals set are achievable. It has been confirmed that the concept developed in the course of this research is effective. However, analysis of the end product shows that the current solution should be treated merely as a starting point – a platform for further extension. Our subsequent goal is to use intelligent data processing automation in order to ensure that the extract, transform and load data process (ETL) is performed in the subsequent stages without the need for any user control – with a virtual, intelligent supervisor deployed only.

Acknowledgements

The research presented in this paper has been conducted as part of an R&D project, co-financed by the EU and by and the Polish Ministry of Digitization, titled: European technological legacy - dissemination of historical and contemporary technical science publications via an innovative IT system, AGH University of Science and Technology, 2016-2019.

References

- [1] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, and J. Widom, "Lore: A database management system for semistructured data", *ACM SIGMOD Rec.*, vol. 26, no. 3, pp. 54–66, 1997 (doi: 10.1145/262762.262770).
- [2] R. Goldman, J. McHugh, and J. Widom, "From semistructured data to XML: Migrating the Lore data model and query language", in *Proc. of the 2nd Int. Worksh. on the Web and Databases WebDB'99*, Philadelphia, PA, USA, 1999 [Online]. Available: <http://infolab.stanford.edu/lore/pubs/xml.pdf>
- [3] J. Shanmugasundaram, K. Tuft, G. He, C. Zhang, D. DeWitt, and J. Naughton, "Relational databases for querying XML documents: Limitations and opportunities", in *Proc. of the 25th Int. Conf. on Very Large Data Bases VLDB'99*, Edinburgh, Scotland, 2008, pp. 302–314 [Online]. Available: <http://www.vldb.org/conf/1999/P31.pdf>
- [4] M. Rys, "XML and relational database management systems: inside Microsoft SQL Server 2005", in *Proc. of the ACM SIGMOD Int. Conf. on Managt of Data*, Baltimore, MD, USA, 2005, pp. 958–962 (doi: 10.1145/1066157.1066301).
- [5] R. Marcjan and J. Wrosteck, "Processing XML documents on the basis of quasi-relational model and SQLxD language", *Studia Informatica*, vol. 32, no. 2A, pp. 111–120, 2011 (doi: 10.21936/si2011_v32.n2A.253).
- [6] N. Nurseitov, M. Paulson, R. Reynolds, and C. Izurieta, "Comparison of JSON and XML data interchange formats: a case study", in *Proc. of the ISCA 22nd Int. Conf. on Comp. Appl. in Indust. and Engin. CAINE 2009*, San Francisco, CA, 2009, USA, 2009, vol. 9, pp. 157–162 [Online]. Available: <https://www.cs.montana.edu/izurieta/pubs/IzurietaCAINE2009.pdf>
- [7] G. Wang, "Improving data transmission in web applications via the translation between XML and JSON", in *Proc. 3rd Int. Conf. on Commun. and Mob. Comput.*, Qingdao, China, 2011, pp. 182–185 (doi: 10.1109/CMC.2011.25).


- [8] M. Piech and R. Marcjan, "A new approach to storing dynamic data in relational databases using JSON", *Computer Science*, vol. 19, no. 1, 2018 (doi: 10.7494/csci.2018.19.1.2505).
- [9] H. Dayani-Fard and I. Jurisica, "Dynamic semi-structured repository for mining software and software-related information", U.S. Patent No. 6,339,776, 2002 [Online]. Available: <https://patents.google.com/patent/CA2284949A1/en>
- [10] V. Christophides, M. Dörr, and I. Fundulaki, "A semantic network approach to semi-structured documents repositories", in *Research and Advanced Technology for Digital Libraries, First European Conference, ECDL'97 Pisa, Italy, September 1-3, 1997 Proceedings*, C. Peters and C. Thanos, Eds. LNCS, vol. 1324, pp. 305–324. Berlin, Heidelberg: Springer, 1997 (doi: 10.1007/BFb0026735).
- [11] D. Tahara, T. Diamond, and D. J. Abadi, "Sinew: a SQL system for multi-structured data", in *Proc. of the ACM SIGMOD Int. Conf. on Manag. of Data*, Snowbird, UT, USA, 2014, pp. 815–826 (doi: 10.1145/2588555.2612183).
- [12] M. Smail-Tabbone, S. Osman, N. Messai, A. Napoli, and M. D. Devignes, "BioRegistry: A structured metadata repository for bioinformatic databases", in *Computational Life Sciences First International Symposium, CompLife 2005, Konstanz, Germany, September 25-27, 2005. Proceedings*, R. Berthold et al., Eds. LNCS, vol. 3695, pp. 46–56. Berlin, Heidelberg: Springer, 2005 (doi: 10.1007/11560500_5).
- [13] D. Tsirogiannis et al., "Scalable analysis platform for semi-structured data", U.S. Patent No. 9,613,068, 2017 [Online]. Available: <https://patents.google.com/patent/US9613068B2/en>
- [14] D. Florescu, "Managing semi-structured data", *Queue*, vol. 3, no. 8, pp. 18–24 2005 (doi: 10.1145/1103822.1103832).
- [15] R. Agrawal et al., "System and method for organizing repositories of semi-structured documents such as email", U.S. Patent No. 6,592,627, 2003 [Online]. Available: <https://patents.google.com/patent/US6592627B1/en>
- [16] D. L. Draper, D. B. Christianson, and K. L. Komissarchik, "Method and apparatus for storing semi-structured data in a structured manner", U.S. Patent No. 6,581,062, 2003 [Online]. Available: <https://patents.google.com/patent/US20060265410>
- [17] J. Komissarchik and E. Komissarchik, "System and method for facts extraction and domain knowledge repository creation from unstructured and semi-structured documents", U.S. Patent No. 8,682,674, 2014 [Online]. Available: <https://patents.google.com/patent/US7756807>
- [18] F. S. Tseng and W. J. Hwung, "An automatic load/extract scheme for XML documents through object-relational repositories", *J. of Syst. and Softw.*, vol. 64, no. 3, pp. 207–218, 2002 (doi: 10.1016/S0164-1212(02)00044-4).
- [19] C. C. Huang and C. M. Kuo, "The transformation and search of semi-structured knowledge in organizations", *J. of Knowl. Manag.*, vol. 7, no. 4, pp. 106–123, 2003 (doi: 10.1108/13673270310492985).
- [20] G. Dobbie, X. Wu, T. W. Ling, and M. L. Lee, "ORA-SS: An Object-Relationship-Attribute Model for Semi-Structured Data", Tech. Rep., School of Computing, Singapore, 2000 [Online]. Available: <https://pdfs.semanticscholar.org/9371/c2ae3e59e2c8b107b39525318ca3ce36c90d.pdf>
- [21] R. Rajugan, T. S. Dillon, E. Chang, and L. Feng, "A layered view model for XML repositories and XML data warehouses", in *Proc. of the 5th Int. Conf. on Comp. and Inform. Technol. CIT'05*, Shanghai, China, 2005, pp. 206–215 (doi: 10.1109/CIT.2005.15).
- [22] L. Liu, C. Pu, W. Han, D. Buttler, and W. Tang, "Building an extensible wrapper repository system: A metadata approach", in *Proc. of the 3d IEEE Comp. Soc. Metadata Conf.*, Bethesda, MD, USA, 1999.
- [23] M. Mani and N. Sundaresan, "System and method for query processing and optimization for XML repositories", U.S. Patent No. 6,654,734, 2003 [Online]. Available: <https://patents.google.com/patent/US6654734B1/en>
- [24] S. E. Madnick and M. D. Siegel, "Query and retrieving semi-structured data from heterogeneous sources by translating structured queries", U.S. Patent No. 6,282,537, 2001 [Online]. Available: <https://patents.google.com/patent/US6282537B1/en>
- [25] D. Skoutas and A. Simitsis, "Ontology-based conceptual design of ETL processes for both structured and semi-structured data", *Int. J. on Semantic Web and Inform. Syst. (IJSWIS)*, vol. 3, no. 4, pp. 1–24, 2007 (doi: 10.4018/jswis.2007100101).
- [26] Metasonic Flow, "User Manual V5.3.5", Metasonic AG Pfaffenhofen [Online]. Available: <https://www.metasonic.de/>
- [27] Doxis4 iECM, "Doxis4 Architecture", Ser Solutions [Online]. Available: <https://www.sergroup.com/en/technology.html>
- [28] M. Piech et al., "Model for dynamic and hierarchical data repository in relational database", *Computer Science*, vol. 19, no. 4, 2018 (doi: 10.7494/csci.2018.19.4.3088).
- [29] J. Han, E. Haihong, G. Le, and J. Du, "Survey on NoSQL database", in *Proc. 6th Int. Conf. on Pervasive Comput. and Appl.*, Port Elizabeth, South Africa, 2011, pp. 363–366 (doi: 10.1109/ICPCA.2011.6106531).
- [30] P. P. S. Chen, "The entity-relationship model – toward a unified view of data", *ACM Trans. on Database Syst. (TODS)*, vol. 1, no. 1, pp. 9–36 1976 (doi: 10.1145/320434.320440).
- [31] K. Y. Whang, B. K. Park, W. S. Han, and Y. K. Lee, "Inverted index storage structure using subindexes and large objects for tight coupling of information retrieval with database management systems", U.S. Patent No. 6,349,308, 2002 [Online]. Available: <https://patents.google.com/patent/US6349308B1/en>
- [32] Z. H. Liu, B. Hammerschmidt, D. McMahon, Y. Liu, and H. J. Chang, "Closing the functional and performance gap between SQL and NoSQL", in *Proc. of the Int. Conf. on Manag. of Data*, San Francisco, CA, USA, 2016, pp. 227–238 (doi: 10.1145/2882903.2903731).
- [33] G. L. S. T. J. Whittaker, "Improving performance of schemaless document storage in PostgreSQL using BSON", CPSC 438 Final Project, April 29, 2013, New Haven, CT [Online]. Available: <https://www.geoffreytitt.com/resources/Postgres-BSON.pdf>
- [34] M. Fowler, "CQRS", Martin Fowler's Blog, 2011 [Online]. Available: <https://martinfowler.com/bliki/CQRS.html>
- [35] DB-Engines Ranking of Search Engines [Online]. Available: <https://db-engines.com/en/ranking/search+engine> (accessed on 2019-09-01)
- [36] N. H. Lim, "PostgreSQL [9.5.0] vs MariaDB [10.1.11] vs MySQL [5.7.0]", 2016 [Online]. Available: <http://nghenglim.github.io/PostgreSQL-9.5.0-vs-MariaDB-10.1.11-vs-MySQL-5.7.0-year-2016> (accessed on 2019-09-01)
- [37] J. Dajda, R. Dębski, M. Kisiel-Dorohinicki and K. Piętak, "Multi-domain data integration for criminal intelligence", in *Man-Machine Interactions 3*, A. Gruca, T. Czachórski, and S. Kozielski, Eds. *Advances in Intelligent Systems and Computing series (AISC)*, vol. 242, p. 345–352. Springer, 2014 (DOI: 10.1007/978-3-319-02309-0_37).
- [38] M. R. Durose, A. D. Cooper, and H. N. Snyder, "Collecting and Processing Multistate Criminal-History Data for Statistical Analysis", US Department of Justice, Office of Justice Programs, Bureau of Justice Statistics, 2019 [Online]. Available: <https://www.bjs.gov/content/pub/pdf/cpmchdsa.pdf>
- [39] A. J. Singer et al., "Victimization, fear of crime, and trust in criminal justice institutions: A cross-national analysis", *Crime & Delinquency*, vol. 65, no. 6, pp. 82–844, 2019 (doi: 10.1177/0011128718787513).
- [40] R. S. Chen et al., "Exploring performance issues for a clinical database organized using an entity-attribute-value representation", *J. of the Amer. Med. Inform. Assoc.*, vol. 7, no. 5, pp. 475–487, 2000 (doi: 10.1136/jamia.2000.0070475).
- [41] P. M. Nadkarni et al., "Organization of heterogeneous scientific data using the EAV/CR representation", *J. of the Amer. Med. Inform. Assoc.*, vol. 6, no. 6, pp. 478–493, 1999 (doi: 10.1136/jamia.1999.0060478).
- [42] O. J. Reichman, M. B. Jones, and M. P. Schildhauer, "Challenges and opportunities of open data in ecology", *Science*, vol. 331, no. 6018, pp. 703–705, 2011 (doi: 10.1126/science.1197962).
- [43] M. Wiener, F. T. Sommer, Z. G. Ives, R. A. Poldrack, and B. Litt, "Enabling an open data ecosystem for the neurosciences", *Neuron*, vol. 92, no. 4, pp. 617–621 2016 (doi: 10.1016/j.neuron.2016.11.009).

- [44] V. Tiwari and R. S. Thakur, "An extended views based big data model toward facilitating electronic health record analytics", in *Telemicine Technologies*, H. D. Jude and V. E. Balas, Eds. Academic Press, 2019, pp. 193–199 (doi: 10.1016/B978-0-12-816948-3.00013-1).



Mateusz Piech is currently pursuing a Ph.D. in Computer Science at AGH University of Science and Technology. His research interests include development of semi-structured data models. More specifically, his work examines different approaches to storing dynamic criminal data in relational databases, which is also

the main topic of his dissertation. He is interested in reactive programming and big data.

 <https://orcid.org/0000-0002-0146-5921>

E-mail: mpiech@agh.edu.pl

AGH University of Science and Technology

Mickiewicza 30

30-059 Cracow, Poland



Bartosz Rakoczy is currently a Research and Testing Assistant at AGH University of Science and Technology. He received his M.Sc. in Computer Science from the same university. His main areas of interest cover proper use of soft skills in IT, team leading and management. His deepest passion, however, is game design. Part

of his work is also focused on forensic science, as well as on storing and analyzing criminal data.

 <https://orcid.org/0000-0002-9324-069X>

E-mail: brakoczy@agh.edu.pl


AGH University of Science and Technology

Mickiewicza 30

30-059 Cracow, Poland



Jacek Dajda currently works at the Department of Computer Science, AGH University of Science and Technology in Cracow. He conducts research concerning information systems (specifically analytical systems for public security applications), software engineering and databases.

 <https://orcid.org/0000-0001-8617-4981>

E-mail: dajda@agh.edu.pl

AGH University of Science and Technology

Mickiewicza 30

30-059 Cracow, Poland



Marek Kisiel-Dorohinicki is currently the Head of the Department of Computer Science at AGH University of Science and Technology. His research focuses on intelligent software systems, and specifically on utilizing agent technology and evolutionary algorithms. He also deals with other soft computing techniques, such as

neural networks or fuzzy systems.

 <https://orcid.org/0000-0002-8459-1877>

E-mail: doroh@agh.edu.pl

AGH University of Science and Technology

Mickiewicza 30

30-059 Cracow, Poland