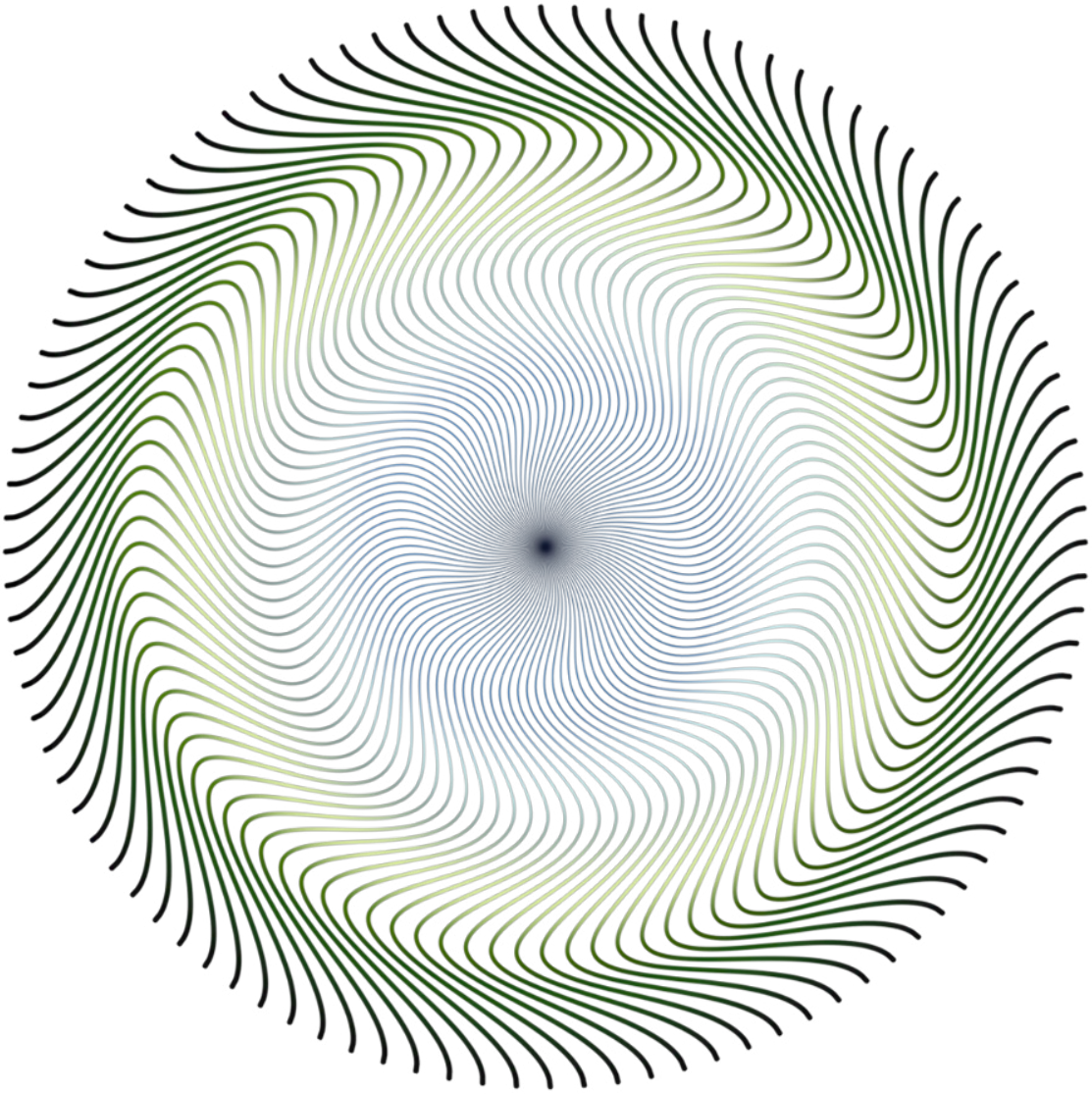
Deloitte.

neo4j



Fraud detection using graph technology

Foreword

Fraudulent applications for COVID-19 bridging grants have a high potential for harm. This whitepaper describes a project aimed at detecting and processing such fraudulent transactions. The procedure for both the creation of the data model and the storage of the data will be presented. This also includes the selection of a suitable solution and the creation of a Proof of Concept (PoC) using graph technology. The approach presented here is also characterized by its reusability.

MAKING AN IMPACT THAT MATTERS since 1845

The project was carried out in cooperation between Deloitte and Neo4j. Neo4j is the leading provider of the graph technology, based in Malmö, Sweden and San Mateo, California. Deloitte is an auditing and consulting firm that provides, among other things, support in the areas of risk and financial consulting as well as services in auditing and tax consulting. The Financial Advisory – Forensic FSI practice, which focuses on the prevention and detection of white-collar crime in the financial industry, was responsible for this project.

Note: The application numbers and key figures presented here are fictitious values which are for illustrative purposes only.

Introduction

The COVID-19 pandemic and the measures to combat it pose major challenges for many companies and self-employed persons. The German government therefore offered financial support to cover their running costs even in the event of closure measures and sales shortfalls. These "Coronavirus aid programmes" were approved and paid out upon application.

The Coronavirus aid programmes were largely transferred to the state development banks. In order to cope with this new task, it was necessary for the state development banks to define new processes.

A large problem with Coronavirus aid programmes were that companies or individuals may have applied for funds without being entitled to do so. This necessitated an accurate examination of the Coronavirus aid applications. At the same time, however, the Corona aid applications had to be processed quickly so that companies and self-employed persons would not run into liquidity problems. Consequently, an intuitive filtering method was needed to speed up the review process. The first step was to filter out conspicuous applications. These could then be subjected to a more detailed examination, while inconspicuous applications were forwarded for rapid payment. Against this background, it was often necessary to check eligibility again at a later date.

Thus, for the implementation of this project (also with a possible focus on retrospective verifiability), on the one hand a catalog of comprehensible criteria was required that could be automatically applied to applications. On the other hand, a user-friendly presentation was needed to support case handlers in the further examination of conspicuous applications.

Technologies – project requirements

Federal and state Coronavirus aid programmes consist of several grant programs. The first grant program used a local process for reviewing and processing applications for emergency assistance. In the following funding programs, a nationwide procedure was subsequently provided by the federal government. Thus, there were different data sources that needed to be evaluated together. Therefore solution had to be able to map both of the aforementioned sources without losing information.

Originally, it was decided to consider the applications individually from each other and to evaluate them using predefined potential fraud factors, but this proved to be incomplete. This meant that overarching anomalies could not be identified, such as the submission of several applications by one person or the receipt of an above-average number of applications from an identical address.

During the processing itself, it was often not clear from the application why it was forwarded to the fraud check in detail. If, for example, an application was manually included in the check due to a name that had previously attracted attention, this contextual knowledge was not transmitted, which made further processing significantly more difficult.

Further allocation and processing of the applications was done via Excel spreadsheets, each of which comprised only a small section of the existing data. These lists were administered via a SharePoint and could only be edited by one person at a time. Manual editing of the lists often led to problems updating the content. In addition, editing was very time-consuming due to the hosting and the volume of the respective data.

Accordingly, the requirements profile of the solution to be developed at that time included the ability to process information from different data sources and to merge them in a networked manner. In addition, cross-application anomalies were to be identified and processed visually and comprehensibly for further examination. Finally, the solution had to provide a stable environment in which several people could make changes simultaneously and quickly.

Description of the approach

How were these requirements implemented? In the following, we would like to briefly explain the essential steps of our approach in this project and point out the most important aspects.

Step 1: Meeting technological requirements

The first step was to find a suitable technology that met the above criteria. Thanks to a long-standing cooperation between Deloitte and Neo4j, positive experience with graph databases was available.

The Neo4j graph database stores data in a labeled property graph (LPG). Graph databases consist of nodes representing specific instances and relations ("edges") connecting them. Nodes and relations are also stored in this form on the hard drive. The former are given labels such as "application" or "company". Relations are labeled by "types" that represent the relationships, such as "SUBMITTED_BY". So, a simple graph could look like this:

(Application) – SUBMITTED BY->(Company)

Properties can then be assigned to the individual nodes and relations so that further data such as company name, application ID, etc. can be stored. A complete image of the graph is called a "data model".

Step 2: Definition of questions and answers

The next step was to plan the implementation of the project. In particular, the previously defined potential fraud criteria for isolated applications had to be expanded to include a higher-level analysis at the relationship level. The existing fraud value represents a weighting of the criteria in this context. For this purpose, additional criteria now had to be defined in order to be able to formulate corresponding queries and optimize the results.

A distinction was made between "soft criteria" (Criteria list A), which can be an indication of a possible suspicion of fraud, but which alone are not sufficient for a comprehensive fraud check and such a check is only carried out if several criteria are present, and "hard criteria" (Criteria list B), which trigger a mandatory fraud check.

Based on the complete list of questions and the associated answers, a decision is then made as to which applications should be subjected to a fraud check. The more criteria are present, the more likely the application is fraudulent. Accordingly, it is then forwarded to different processing teams.

In addition to the existing criteria, further criteria have been added based on the relationships between different applications, such as:

- Has a natural person in the function as a self-employed person or as a person authorized to represent a company submitted several applications?
- Have further applications of the self-employed person or company been classified as conspicuous in the past?

Step 3: Derivation of the data model
A data model was derived from the collected criteria in collaboration with the experts from Neo4j. In the process, the original model was adapted and optimized in several iteration cycles to better answer individual question criteria or to support additional questions with the model. The revised data model for our COVID-19 fraud project is depicted in Fig. 1.

In the graphical representation of the data model, the extracted entities (nodes) such as application, company, person, IBAN block list, check, criteria list A (for soft criteria), etc. can be obtained. Since the information is stored as separate nodes, repetitive information can be consolidated across funding programs, simplifying further analysis. Such repetitive information includes, for example, addresses that have been used more frequently or companies that have submitted multiple applications.

The individual entities are then related to each other, represented by the arrows (relations) between each node. The information is stored and queried in this form in the graph database which also allows, for example, different applications to be attributed to one entity.

What the diagram does not show are the "properties" of the individual nodes and relations in the graph. These properties represent data fields of the individual nodes. They can store data about the application, such as the application ID, amounts applied for or paid out, and other important information. These can also be used for further analyses and visualizations.

Step 4: Preparing and loading data
After the successful development of an initial data model, data can be loaded into the graph database. In the context of the proof of concept, this step was initially carried out on a test basis using various Excel tables. To protect data privacy, only pseudonymized data was used here. These were generated in advance by an internal software solution.

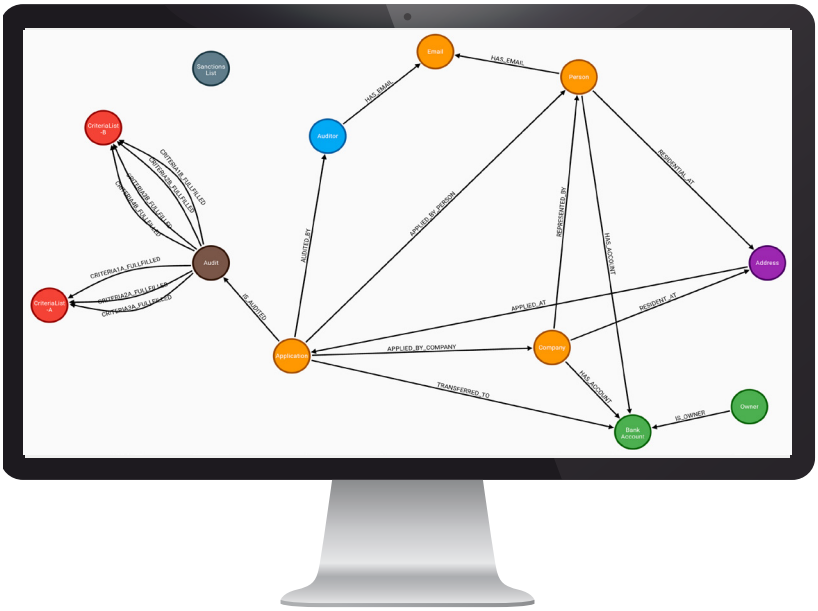
Overall, the data import can be further optimized at this stage. The Neo4j query language Cypher, which can be described as the SQL of the graph database world, provides various functions for this purpose. For example, formats such as CSV files, but also JSON or XML data can be read in. Access to databases via standard interfaces such as JDBC or ODBC is also possible. If a customer already uses a ETL tool (Extract, Transform, Load), this can also be integrated to load the graph.

In the context of the Coronavirus aid programmes, the number of grant applications increased daily. Thus, new data sets were generated permanently. Here, too, an interface solution would lend itself to ensuring live access to the built-up database.

Step 5: Data quality management and fuzzy logic
When loading the pseudonymized data, a quality check revealed problems due to deviating spellings. When comparing different applications with regard to addresses, company names and natural persons, there were often different spellings in different applications. This made it difficult to identify potentially fraudulent applications.

To address this problem, text fields such as company and individual names were indexed using a full-text index. This mechanism examines texts in the style of a search engine and then offers "fuzzy logic" for the search. This means that operators can be used to search for similar names. The closer the result is to the search expression, the more prominently it appears in the results list. A special score is calculated and additionally shows how much expressions resemble each other. This is a step towards entity resolution, which involves improving data quality by identifying duplicates and related data records.

Fig. 1 – Data model for COVID-19 fraud detection (without properties)



Step 6: Answering questions using query language or visualization tools

Once the data has been loaded and checked using the entity resolution measures, data can be queried using various graphs. This can now be used to check whether the list of questions (see step 2) can be answered, and an evaluation can be performed. The query language Cypher and other visualization tools can be used for this purpose. In this project, the tools NeoDash and Neo4j Bloom proved to be suitable.

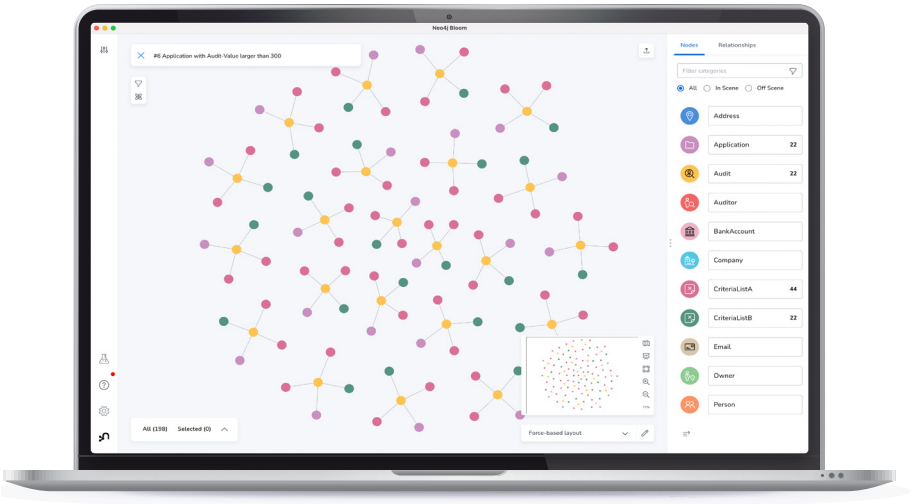
In a first step, Neo4j Bloom can be used to check which applications have fulfilled which predefined criteria. This verifies the original data to also ensure the quality of the system set up. Since additional potential fraud criteria have been added through this process, search queries can also detect more potentially fraudulent applications than is the case with the original data set. Nevertheless, it can be determined whether the identified potential fraud cases here are congruent with the original applications. For this purpose, the results obtained by this procedure can be downloaded as an Excel file and then compared with the original data.

However, this view can be used not only to display requests for the overarching identification of potentially fraudulent requests. Neo4j Bloom can also be used in direct processing to filter for individual requests and display all potential fraud factors as well as properties and associated nodes. In addition, simultaneous processing of individual properties of applications is possible without causing time delays.

The advantages of using Neo4j Bloom are the ease of searching and the visual analysis of the data sets stored in the graph (the database). This allows clerks to use the tool without having to be familiar with the data model or the stored data fields.

In addition to this, Neo4j Bloom can be extended to include queries in natural language (NLP, Natural Language Processing). This allows case workers to use text sentences when analyzing data, for example,

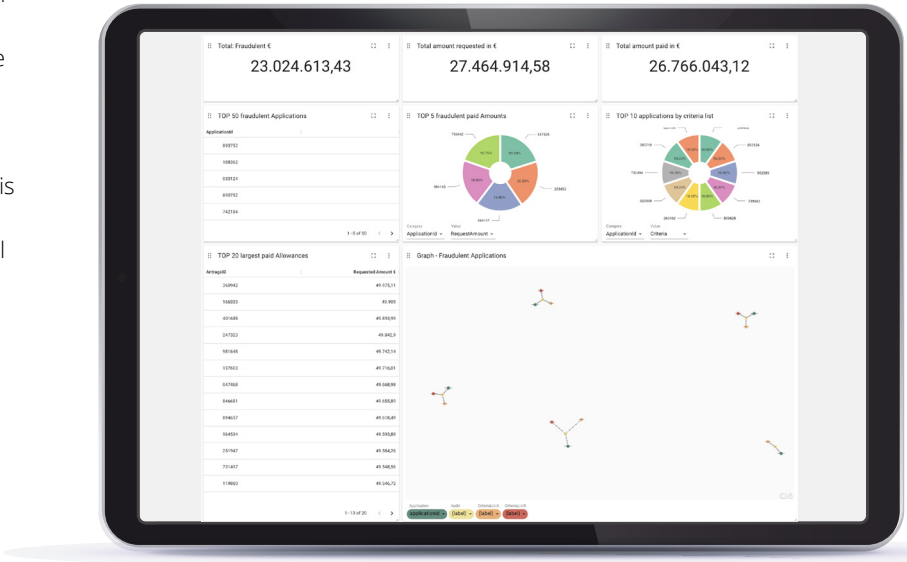
Fig. 2 – Neo4j Bloom – analyzing evaluation of fraudulent data



"Show me all applications from company X with a fraud value greater than Y." These NLP queries can also use dynamically generated search values in the text query (example: X as a fuzzy search of a company name, Y as a fraud value). This increases the flexibility of use.

As another tool, NeoDash can be used within the elaborated solution to optimize the reporting of applications that are currently being processed. This is a data visualization software for Neo4j, with which individual dashboards can be created. These then provide, for example, an overview of the current project progress or pending applications, totals, etc. NeoDash uses real-time data from Neo4j for this purpose. Figure 3 provides an exemplary representation of such a report.

Fig. 3 - Neodash – key figure dashboards



Further steps: Revision, extension and additional development

Additional adjustments will be made in due time, such as revising the data model and the queries if additional requirements are added, or if further improvements to the queries are needed. These further optimization steps will be described in more detail at a later stage.

In addition, the automation of the processes will be further developed and optimized (also with regard to other projects). This concerns both the loading of data using the ETL pipeline (Extract, Transform, Load) and the tool used by the customer, as well as the automated screening of new data. Through the latter, even early detection can be realized. In this way, the entire work process for processing is made much easier and equal treatment in the processing of applications is ensured.

Depending on the customer's environment and the way a solution is operated, additional steps may also be required – for example, for commissioning, operational support and monitoring of the solution.

These points require customer-specific individual implementation, which is why they cannot be discussed further within this paper either.

Transferability of the solution

The approach to building graph databases tends to be similar. The solution presented here provides a good example of potential fraud use cases and can be reused in parts for other projects. In general, however, building a data model is easy and intuitive to implement if the appropriate expertise (domain knowledge) is available. From the analysis of the problem to the first query often it takes only a few days.

In this project, a graph database, also called a "knowledge graph" (KG), was created for potential fraud detection in Coronavirus aid applications. The KG then contains the knowledge base for one or even more use cases.

Once the graph has been built, the task is to develop it further in line with requirements, thus providing even more extensive support for the customer's business processes.

Other areas of application besides the prevention of potentially fraudulent transactions include, for example, other areas of risk management. The graph can be used as a knowledge base for the data science department, which can further improve the approach with its methods.

Summary

The Proof of Concept for the detection of potential fraud in Corona aid applications shows the strengths of graph technology.

Especially in use cases with high complexity or a large amount of linked data, processing with means like Excel or simply relational databases is often not target-oriented. Here, the graph database approach – as shown – offers a forward-looking solution.

Contacts

Matthias Rode
Partner | Financial Advisory FSI
Tel: +49 151 58002270
mattrode@deloitte.de

Dr. Christoph Wronka
Director | Financial Advisory FSI
Tel: +49 69 75695 6037
cwronka@deloitte.de

Janina Uspelkat
Senior Consultant | Financial Advisory FSI
Tel: +49 40 32080 4908
juspelkat@deloitte.de

Deloitte.

Issue 11/2022