Universiteit Leiden
The Netherlands

BSc Computer Science & Economics

An AI based approach to Customer Data Cleaning

A Case Study in Master Data Management at Nationale-Nederlanden

Kylian Kropf

1st supervisor: Dr. G.J. Ramackers
2nd supervisor: Prof.dr.ir. J.M.W. Visser
Company supervisor: A.E. Boukema M.Sc

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS) Jan - Aug 2020

Abstract

The growth of databases and centralizing data from various sources has contributed to data quality problems by increasing the number of incorrectly entered and outdated data in many public and private organisations. In order to clean and restructure a database, manual data cleaning is often used, which is a costly and time consuming process. This study developed an AI based automated solution to this problem.

The work is based on a case study of the manual cleaning process of the customer master database at NationaleNederlanden insurances. It provides a detailed analysis of the manual data cleaning process, and proposes an AI based implementation that augments, and minimises the manual process.

An initial Rule based web app has been built which covers the manual steps performed by employees. Both internal data, and external data from the Dutch Chamber of Commerce are combined, leaving interpretation and decision on what to do with the data to the employees. Basic situations (10.17%) can be solved automatically by this rule-based application. While using the application, the employees' actions and decisions of the remaining records are recorded to create a training set.

These recorded human decisions are subsequently used to train a Logistic Regression algorithm. This combined Rule-based and trained Regression algorithm is then used to clean most of the database records automatically.

By adding the Machine Learning component, an estimated 78.4% of the records could be cleaned automatically, whilst maintaining quality requirements. This resulted in potential estimated time savings of 85.3% and cost savings of €2.161.000 on the customer data cleaning project at NationaleNederlanden.

Acknowledgements

The completion of this research could not have been possible without the assistance of many people whose names may not all be enumerated. Their contributions are sincerely appreciated. I would like to express my sincere appreciation and indebtedness, particularly to the following:

Guus Ramackers as first supervisor for inspiring my interest in the development of innovative technologies and his dedication for this research. Second supervisor, Joost Visser who gave constructive criticism which led to exciting new insights.

Anouk Boukema, must be thanked for her guidance and impressive involvement through each stage of the process. Lisanne Parre deserves many thanks for her remarkable leadership on the entire team. I am very grateful for the support of the technical department during the setup of the development environment and accelerating the safety procedures to speed up the process.

Both NN and LIACS turned out to be very flexible during these outstanding times. Without this, it would not have been possible to finish this research on-time.

Contents

1 Introduction

1.1 Problem statement

In most organizations, one database is used as a primary database, also called the main (or master) database. All supporting flows and processes like billing, mail and CRM (Customer Relation Management) software are directly or indirectly connected to this database. The main advantage is that only one system needs to stay up to date to maintain data integrity. The supporting systems mirror the data from the main database. Many organizations are struggling to get and maintain a clean database. Dirty database records can have many sources, for example: duplicate values, missing values, typos, phonetic errors, non-existing addresses or contradicting values.

Data cleaning (also called data cleansing) is the process of detecting and correcting (or removing) corrupt or inaccurate records from a database. Inaccurate data does have serious real-world implications.

Due to recent developments concerning money laundering and preventing fraud, it has become even more critical to clean the main database. In the Netherlands, financial institutions are required to follow the rules from the Wwft (Money Laundering and Terrorist Financing Prevention Act) [1]. This law is enforced by the Nederlandsche Bank [2].

The challenge for this research is finding a method of automatically grouping database entries on a real-world entity registered in the Chamber of Commerce.

1.2 Goal of this research

Although inputting the right customer data seems simple at first glance, it is not. Information is placed in the database by untrained employees or even by customers themselves. Literature on how to prevent new manual false inputs is already available and therefore, out of scope for this thesis. In past competitor acquisitions databases were merged which contributed to the data integrity issue. Future company take overs will keep adding to this; hence a long term solution is required. The results of the new approach should be at least be the same quality as the current manual process. Lower values would result in customers seeing not their own data and are therefore not permitted.

The main goal of this research is to build an automated model of verifying all records resulting in a significant reduction in time spent on cleaning the database. The tool will use AI to check the records and show relevant and probably right inputs in an explainable way to the data cleaning team. Besides the primary goal, this thesis will be looking at overall process improvements and the financial aspects of speeding up the cleaning process.

1.3 Research questions

At Nationale-Nederlanden, database cleaning is an important subject. To get insights into the people behind a company (the beneficial owners), to comply with the Wwft, a data cleaning project is initiated. This means all business customers are inspected manually. The goal stated in section 1.2 leads to the following research question:

RQ: To what extend can the data cleaning process be improved by implementing AI?

To answer this question, some way of measuring improvements must be developed. For this we have a sub-question:

RQ.a: How can the quality of data cleaning by measured after the implementation of AI?

Next to AI, there might be additional automation steps necessary to improve the quality of the process. This sub-question is related to this issue:

RQ.b: Is it possible to automate steps from the process on a rule base?

We will also study whether AI actually improves the quality using our measurement method developed in RQ.a.

RQ.c: Can the improvement after implementing AI be measured in practice?

In order to get insights in the financial aspects of transforming the process, the following question have been setup:

RQ.d: What is the Return on Investment (ROI) of implementing AI in the data cleaning process?

Research question a, b, c and d will be answered through a case study at Nationale-Nederlanden.

1.4 Nationale-Nederlanden

This project is a collaboration between Nationale-Nederlanden (NN) and the Leiden Institute of Advanced Computer Science (LIACS). NN is one of the biggest financial service providers in the Netherlands. A big part of NN customers consists of businesses. All of these businesses their data need to be entered correctly into NN's SAP BuisnessPartner's (SAP BP) database, which is the main database. All business processes, like billing, payouts or product overview on the online business portal, are connected to this database. NN has several different business units; each unit is responsible for a specific insurance package. Overarching these units is the C&C (Customer and Commerce) unit, they are responsible for the cooperation between the business units and therefore responsible for the databases.

To get a clear customer profile, both for legal and simplicity reasons the C&C department merged all the client data from all the business units and company take overs into one database. This

database is shared with all business units. There are approximately 250.000 business entries who that all have to be verified manually for duplicate entities and false values. When the cleaning is done, the clients will be provided a digital portal for faster, easier and cheaper support.

To see if a business entry in the database is correct, the Data Quality team has to verify the record with the customer's information on the website of the Kamer van Koophandel (Dutch Chamber of Commerce) and internal data from the SAP BP database. To speed up the process and prevent human failure a project was initiated to drastically speed up the process with the help of automation and AI.

2 Research approach

During this project, we adapted the research-based design methodology [4]. This process is oriented towards the building of prototypes, testing and redesigning the solution.

The first stage focused on understanding the problems, environment and culture. The results of this are better understanding the context by identifying challenges and opportunities. Informal interviews with the employees have been conducted, sketches of the process have been made and source code inspected. The results have been reviewed by the employees to prevent noise. Next, a focused review of the literature benchmarking existing solutions are made in order to develop insights into possible challenges. In the design stage, the knowledge from the previous steps is used to design a solution. After a session with the employees, an approach is chosen, and software built. Continuing, in recurring sessions, feedback will be gathered to improve the software. Finally, the results are validated to make sure the new approach has better results.

3 Existing situation

This chapter describes the as-is cleaning situation at NN and definition of terminology applicable in the NN customer data cleansing context.

3.1 Data Cleaning team

The data cleaning team is part of the task force data quality within the company. The team consists of 20 FTE, who are hired externally. Next to the data cleaning team, a team within the task force is dedicated to preventing new faulty inputs into the database.

3.2 Problems with the data

At NN, incorrect entities show up at 46% of the customers in the main database. Problems can have many sources; the most common problems are:

- The same company is registered multiple times in the database
- Chamber of Commerce-numbers are incorrect

- The business is de-registered from the Chamber of Commerce
- The company restarted under a different Chamber of Commerce-number
- Multiple companies registered with the same name in the database

3.3 Tried improvements

In the past years, the company already tried to solve the problems with a rule cleaning approach. Only the de-duplication software has been found working. The de-duplication software only merged data when it was over a pre-set threshold, the remaining records remained ungrouped. Due to the complexity and entanglement of the problems, an incremental automation approach did not work.

3.4 Database design

The SAP main database is centred around Golden Records, which represents legal entities. All legal entities in the Netherlands, so even companies who are not a customer, are present in the database as a Golden Record. The values within a Golden Record are copied from the most recent edited Silver Record. Insurance policies are held within a Silver Record which is always related to a Golden Record, as a parent-child relationship. Silver Records are the inputs from the Business Units. Silver Records can include payment information. The UBO's (Ultimate Beneficial Owner), as well as a Chamber of Commerce record, are Silver Records which are connected to the Golden Record (as a legal entity). UBO's and Chamber of Commerce records are pulled automatically pulled from the Chamber of Commerce and do not hold products.

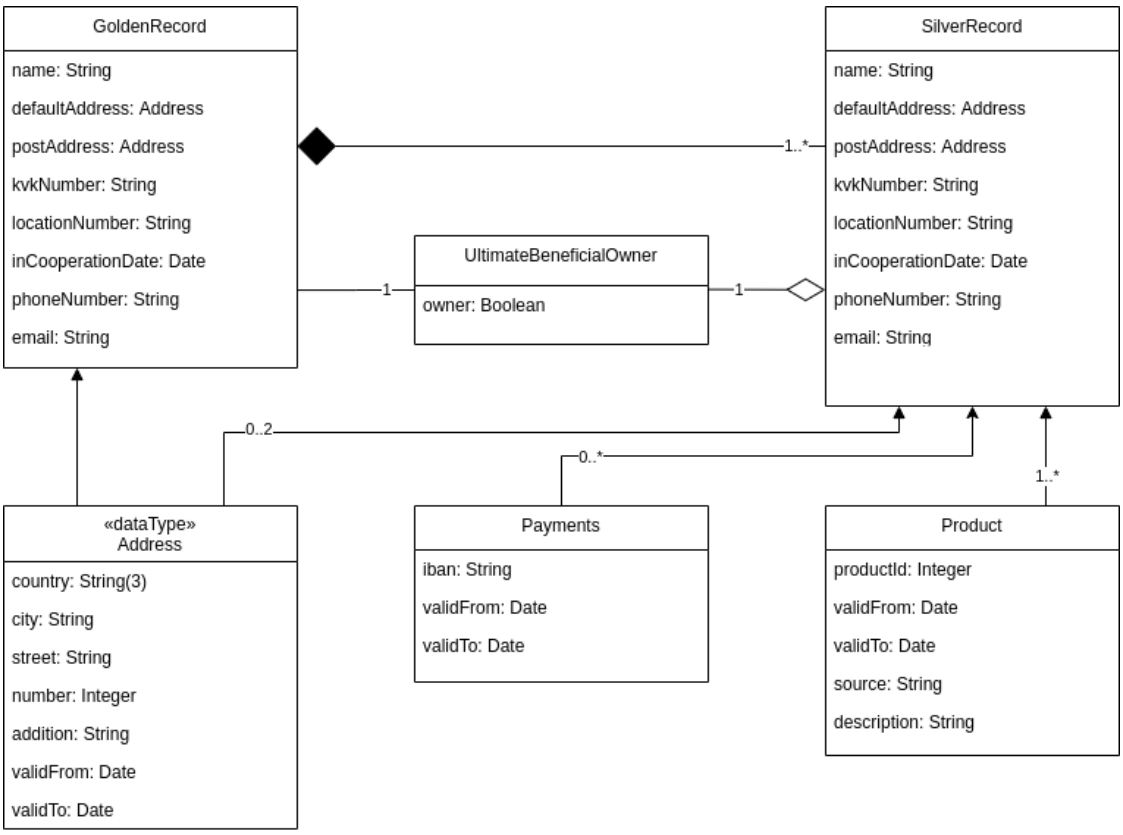


Figure 1: UML class diagram [3] of core customer elements.

3.5 De-duplication software

The SAP database connects to a tool from Human Inference (HI). This program generates a Golden Record (GR) where a Silver Record (SR) connects on. The goal of the program is to connect SR to a GR as a parent-child relationship. The HI software consists of three main components: name standardization, address standardization and de-duplication.

3.5.1 Name standardization

The name standardization is performed when a business partner is modified/created. This is done automatically in the background. Name values are not always appropriately formatted or placed in the correct field name when arriving in SAP. HI can improve the quality of the company name. Names are converted to a capital letter followed by lower case letters abbreviations are fully converted to capital letters. For example, abc Consulting is converted to ABC Consultancy.

3.5.2 Address standardization

The software can improve the quality of an incoming address (applicable on Dutch addresses). All addresses should be formatted the same way to make searching possible. The address fields are passed to the HI software in which they are normalized. It is capable of completing not filled in addresses. For this, it is required that the address is identifiable. If the zip code, house number and the street with the wrong city are entered, this is considered as multiple possibilities. It is an

exception when the city is very similar to the correct city. For example, city names that occur more often, like Vianen ZH and Vianen UT. When filling in the zip code 4132 VE house number 4 and the city Vianen ZH, HI will then convert this to Vianen UT based on the zip code.

3.5.3 De-duplication

It is checked whether the business partner can be linked to an existing GR. Before a GR can be determined, a valid address and organization name has to be filled in. For persons also the initials have to be filled in. A valid address is determined as follows:

1. For every administrated address type, only the address with the highest-end date is used when searching
2. An address must have the combination of a postal code/house number or postal code/ PO box, otherwise, the address is not used for the HI search.

After determining the address, the other mandatory fields for the HI search are checked. For each HI search executed, it is checked whether there is a HI result with a score above the threshold. If such a score is found, no further searches are executed. Older Silver Records can contain just a name or just a name and city and will therefore not be found.

All the results with a score below the highest score are removed from the result set; the scores below the threshold are also removed. If more than one result record remains, then the first one will be used, and a merge with the remaining records is necessary.

If both the source record and the HI result have a Chamber of Commerce number, both numbers have to be equal. If this is not the case, then the HI result cannot be used and will be removed from the result list. Next, the Chamber of Commerce number and establishment number have to be equal, if not the result cannot be used. If the establishment number is equal, then the HI score, if this score is below the automatic border, will be incremented to exactly the threshold. This will cause a de-duplication, even when the initial score is too low.

If a golden record is already linked to a Silver Record and changes on the Silver Records take place by the business unit on critical areas, then a split has to be executed. For example, the Chamber of Commerce number has changed, the company may be another than the linked golden record.

3.6 Current cleaning process

The current cleaning process is built around an Excel sheet that has to be filled in to force the employees into taking certain steps. The goal of this process is to connect a Silver Record to a real-world legal entity (Golden Record). Problems, like incorrect or missing data, with the Silver Record are solved manually during this process. Employees manually search in the SAP database for Silver Records that have a connection with the current golden record. They use the Chamber of commerce website to check the legal status and official address of the legal entity. Given a Golden Record number, these questions are:

- Company has only one department/establishment?

- Golden Record has Chamber of Commerce number?
- Chamber of Commerce number valid?
- Are there Silver Records with other company names?
- Are there Silver Records with other addresses?
- Chamber of Commerce number occurs more often in main database?
- Company name occurs more often in main database?
- Address occurs more often in main database?

Based on these questions, the Golden Record can be categories in one of the following four statuses:

Status	Variables to take in consideration
Archived	• Golden Record does not hold Silver Records
BU assigned	• Unidentifiable Silver Record has been found in the process [1] • Company de-registered from Chamber of commerce and still has active products under the Silver Record(s)
Stopped	• The company related to this record has stopped, and it does not have active products.
Finished	• Company de-registered from Chamber of commerce and no active products under the Silver Record(s) • Additional Silver Records has been added to golden record

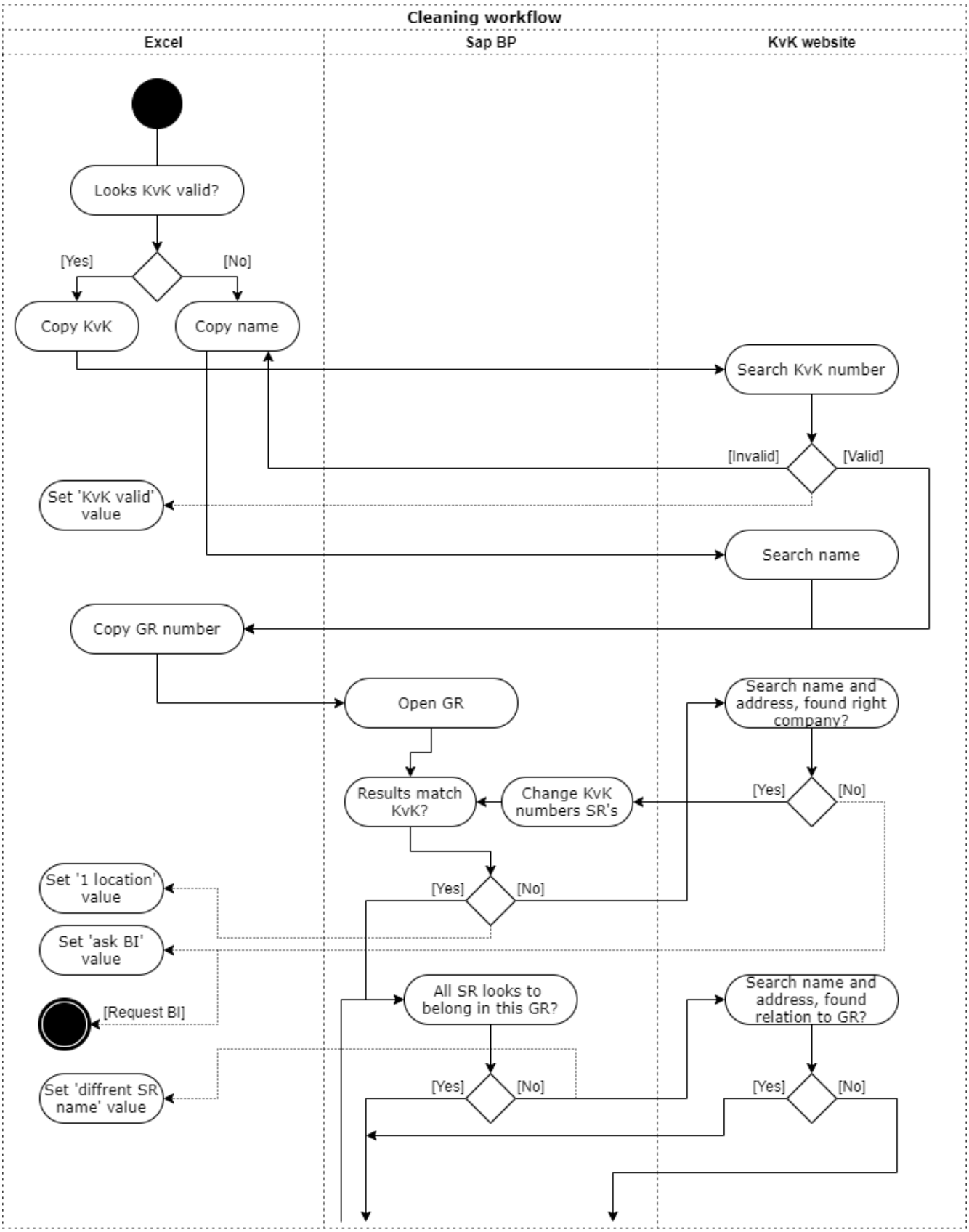
Table 1: final status of an inspected Golden Record

After the golden records has been assigned to one of the categories, additional steps have to be made to finish the golden record.

If the status is archived or stopped, the golden record will be removed automatically after the legal retention period has ended. If a Business Unit has been assigned, there is not enough information available and the BU has to contact the customer to solve the problem. Finally, if the status is finished, the golden record has been checked. Wrong Silver Records are removed and additional Silver Records are manually added. Now the golden records should to be flagged as 'checked' in SAP. This ensures that this record will stay 'clean', which makes it not possible to remove Silver Records from the golden record.

A process description of the current process outcomes of the results from Chamber of Commerce and the main database is provided in figure 2.

[1] Always the result of manual error on inserting. Example: ABC Accounting Holding Ltd. and ABC Accounting Group Ltd. are registered at the Chamber of commerce. ABC Accounting is registered as a company SAP, it is impossible to determine to which legal entity the record is part of.


Cleaning workflow
Excel
Sap BP
KvK website
Looks KvK valid?
[Yes]
[No]
Copy KvK
Copy name
Search KvK number
[Invalid]
[Valid]
Set 'KvK valid' value
Search name
Copy GR number
Open GR
Search name and address, found right company?
Results match KvK?
Change KvK numbers SR's
[Yes]
[No]
Set '1 location' value
Set 'ask BI' value
[Request BI]
All SR looks to belong in this GR?
Search name and address, found relation to GR?
Set 'diffrent SR name' value

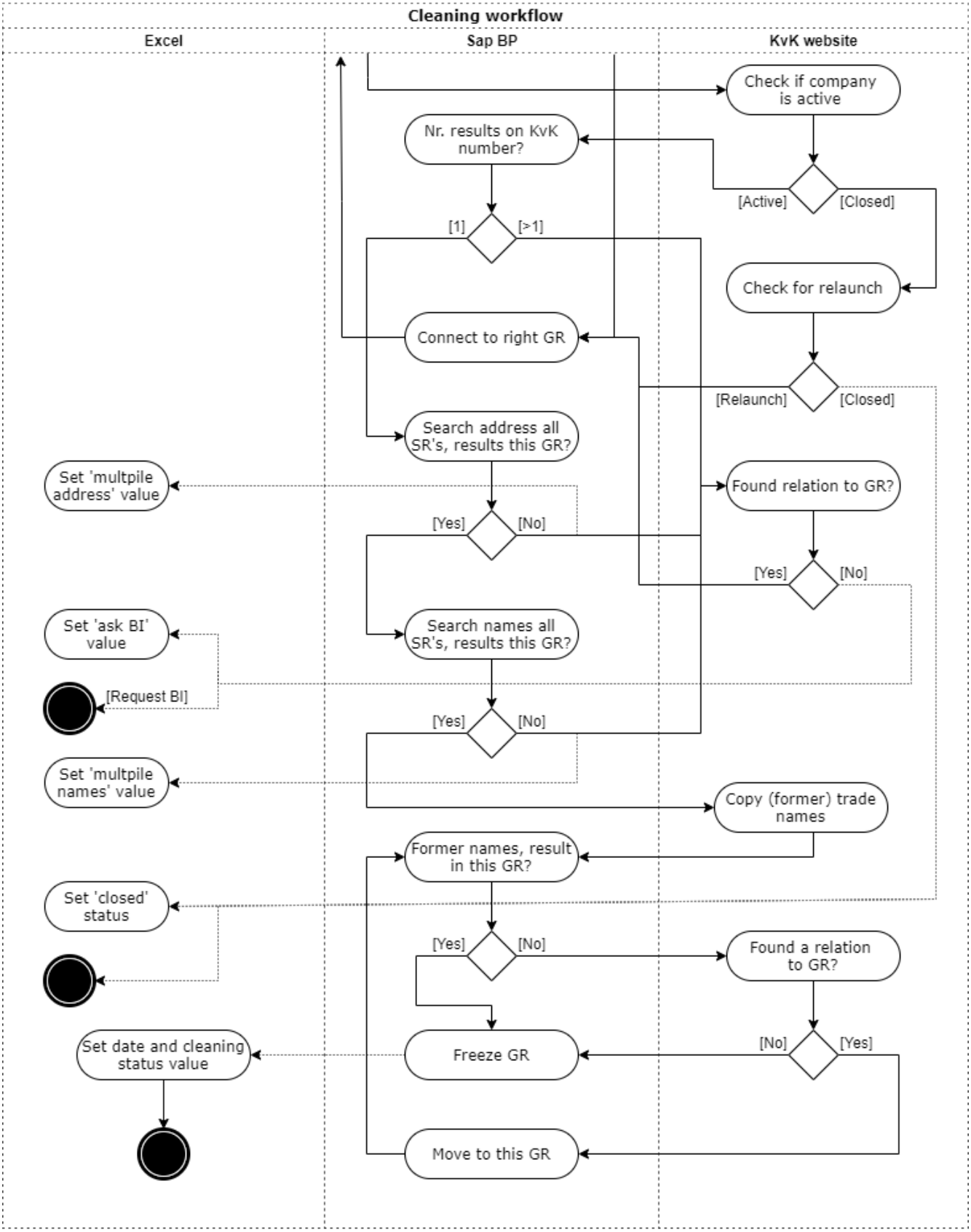


Figure 2: manual cleaning process without automation

The employee compares the SAP BP data with the Chamber of Commerce data in a structured way. In the table below all branches in the current process are displayed.

Source	Action	Output
SAP	KvK from GR non-real	true / false
KvK	KvK from GR active	true / false
KvK	Branch number from GR active	true / false
KvK	Does KvK from GR have multiple locations	true / false
SAP	Should the SR's be under this GR	if no: move manually
SAP	More records on address of SR's	if yes: check manually
KvK	More records on address of SR's	if yes: check manually
SAP	>1 results on (similar) company names	if yes: check manually
KvK	Results for former trade names in SAP BP	if yes: check manually

Table 2: decision variables to connect a Silver Record to a Golden Record

3.6.1 Cons of current manual approach

Cleaning a record becomes more and more time consuming with each problematic entry, this leads to more time than necessary being allocated for cleaning the database. Next to this, the manual steps increase the chance of a manual error showing up.

4 Overview of existing cleaning approaches

In this chapter, a systematic review of possibilities for solving main data issues will be addressed. Data quality problems are as old as the first databases, most data quality literature has been written with the technical knowledge of the nineties.

4.1 Preventing false inputs

Literature on how to prevent faulty input in a main database in the first place has been available for many years. Research showed the most important factor in building and, more importantly, keeping a clean database is responsible employees [5] who take ownership of the data [6]. Like mentioned earlier this research will not dive deep in the subject of preventing false inputs.

4.2 Types of data quality problems

Although data quality problems have many sources, the problems can be classified in several groups [7].

- Single-source problems are related to field sources without a schema; there is a lack of restrictions on what data can be entered, resulting in a higher chance of errors. This allows illegal values, misspellings, duplicates and different values.
- Multi-source problems occur when single sources are aggregated. This can cause naming conflicts to show up. Naming conflicts are present when the same name or description is used

for different objects. Also, synonyms or abbreviations could be used. Even when records have the same name, there might be a difference in values (for example, new and old address).

Single-source problems can be solved relatively easily by applying checks and rules to the (new) data, for example: a KvK number should only contain numbers and has a length of 8. Every value which does not meet this requirement can be ignored. Multi-source problems can be solved by identifying overlapping data, more particularly data which refer to the same real-world entity (also called duplicate elimination [8] or object identity problem [9]). If the values are partially wrong, it might be possible to correct the wrong value(s). If this is not possible, this information should be gathered manually; for example, by calling the customer to verify his details.

4.2.1 Approximately duplicate entities

If entities are duplicates, the degree of similarity (or closeness) of the two entities can be calculated [14]. Most approximate match predicates return a score between 0 and 1, where 1 being assigned to identical entities. An approximate match predicate will consist of two parts [13]:

1. Atomic Similarity Measures: calculates the edit distance and phonetic distance.
2. Algorithms to combine similarity measures: based on a set of pairs of attributes belonging to two entities, in which each pair is tagged with it' s own approximate match score.

A variety of record linage algorithms have been developed and deployed successfully. However, the set of parameters have to be set by field experts and altered during the execution. Finding the ideal values of such parameters is not straightforward; most of the time, the ideal threshold value does not exist, as it practically can not be found. Research found this could only be solved by dynamically changing the parameters of the record linkage algorithm [15].

4.3 Automated cleaning

A incremental approach is usually taken in data quality problems. Big problems will be solved first, working down all problems in the entire database. In order to prevent new errors, the incremental approach requires a structured approach:

1. Data analysis: detect the kind of inconsistencies and errors.
2. Transformation and mapping rules: describe the technical rules to solve the problems.
3. Verification: verify the corrected and effectiveness of the rules on a sample of the data.
4. Transformation: execution of the rules on the entire dataset.

There are three approaches for the data analysing step.

- The metadata approach [10][11] looks at metadata for finding data quality problems. It can identify attribute correspondences, for example: false inputs made by the same person.
- The data profiling example focuses on the instance analysis of individual attributes; it derives values like data type length and value range (for example: KvK number example above).

- A data mining approach helps to discover data patterns in large data sets, for example a parent-child relationship. The data mining module includes clustering, memorisation, association discovery and sequence discovery [8]. Integrity constraints among attributes such as functional dependencies or business rules, can be derived [12]. This can be used to correct illegal values and identify duplicate records across data sources. For example: an association rule with high confidence can emphasise data which does not follow this rule. With a confidence of 99% the rule "length(zip)=6" applies, this indicates that 1% of the records do not follow this rule and require closer examination.

5 Design and implementation of a new approach

The new design relies on two steps of automation. In the first step, all the manual steps of the process are removed and replaced by the program. The Excel sheet and SAP BP systems are no longer necessary. Some steps are automated on a rule base; the final decision to connect a Silver Record (child) to a Golden Record (parent) object is still made by an employee. In the second step, Artificial Intelligence is used to determine if Silver Records should be connected to a Golden Record.

5.1 Human supportive software

The first step in speeding up the process and preventing human error is building a tool which shows the available information from the SAP BP database and the Chamber of Commerce in a structured way to the employee. By doing so repeated manual steps are removed, and possible matching records will be looked up automatically. Based on table 2, which holds a maximum of 5 manual human interpretation points, the following scheme as new architecture has been made. The grey blocks match the human decision points.

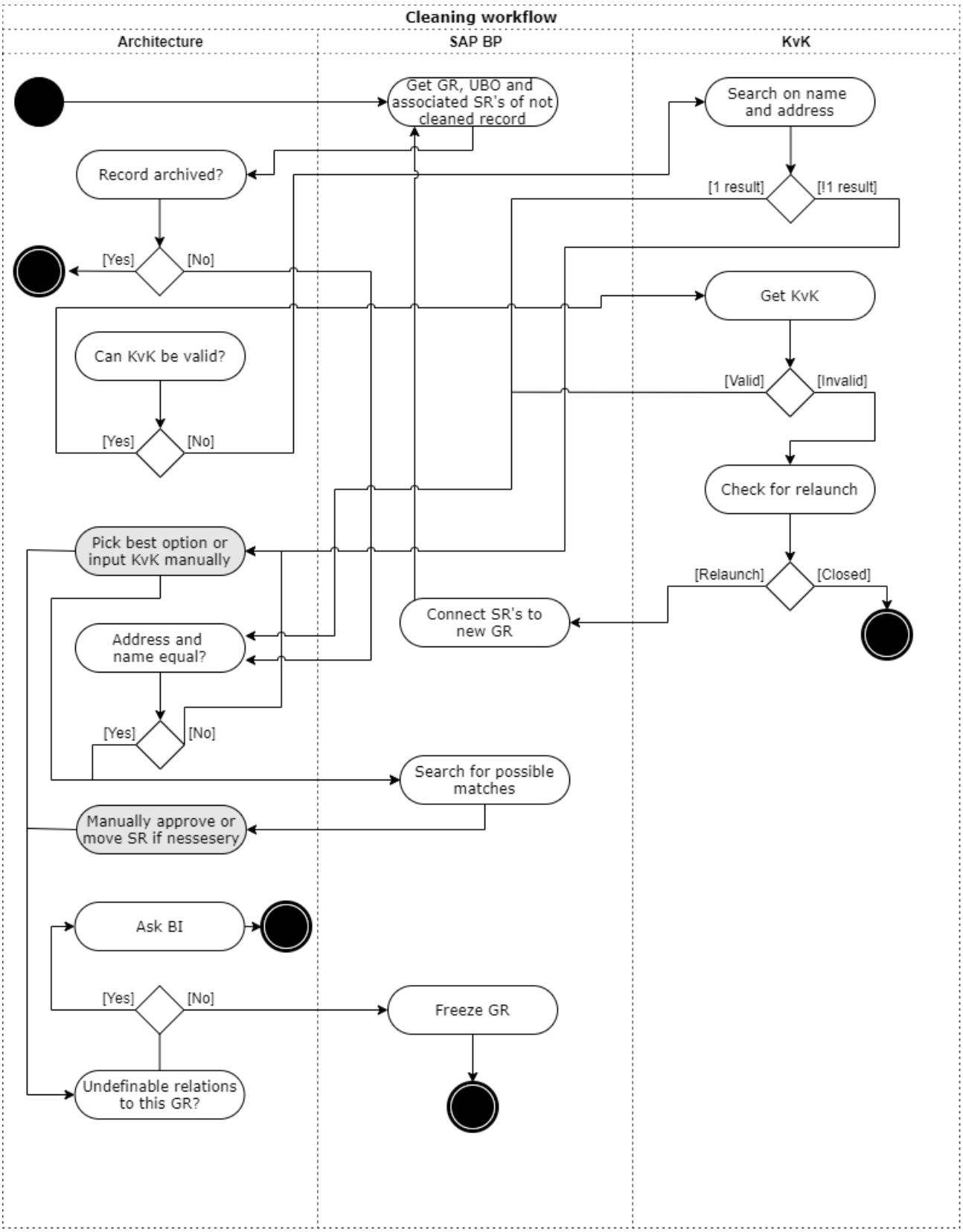


Figure 3: design of a new cleaning approach

5.1.1 Back-end

As the back-end is made for integration with the NationaleNederlanden IT infrastructure, there might be some heterogeneous integration issues. For example: no direct write access to SAP BP, inconvenient database design or manual Golden Record number input in the frond-end. To make the program easily accessible, allow low-technical knowledge and support fast deployment, a web app program has been built. The back-end runs on PHP, as there is no performance issue.

To protect sensitive data, the production environment has been set up in Microsoft Azure, allowing only specific IP ranges to prevent 3rd parties from inspecting the program. Users can only access the database with a personal temporary hour token. This token only has permission for the required database tables. All the database actions are logged and closely monitored to prevent abuse.

The steps of the back-end:

1. Given a Golden Record number, the current Silver Records (insurance policies) are collected. If the Golden Record does not have any active Silver Records, this Golden Record can be ignored.

2. The Chamber of Commerce field is not required while inserting so might not be filled in. Filled in at the Silver Records can not be trusted and are ignored by the back-end. Even when the Golden Record contains a Chamber of Commerce Silver Record, this will be ignored completely. Based on the company name and address in the Silver Record, possible KvK registrations will be pulled. If only one result has been found, this KvK record (and connected Golden Record) will be used in step 3. If no or multiple KvK registrations are found, an employee should manually pick a KvK registration first.

3. A list of all trade names, former trade names and statutory names is downloaded from the KvK API and added to the KvK object. A fee should be paid to the KvK for this action.

4. If available, the UBO details are pulled from SAP BP.

5. Silver Records which have a connection with the KvK record, current Silver Records and UBO (if available) are searched. Possible related Silver Records will be searched on: KvK number, addresses, the normalized version of all (former) trade names, phone numbers, and website domain. Inactive Silver Records are ignored.

6. The result is an array with: current Silver Records, KvK record, UBO record, possible matching Silver Records. This result is printed as a JSON object to be processed by the front-end.

To cope with different standards of notation and personal preferences to abbreviate a company name, a normalize function has been set up. Abbreviations, company structures as well as other standard terms, are removed to find all possible matches. For example, a Silver Record with the name 'Board of Example Company Ltd.' will be 'Example Company'. For example, it might be possible that this company is also registered in a Silver Record as 'Example Company Inc.' or 'Example Company Inc. in Amsterdam'.

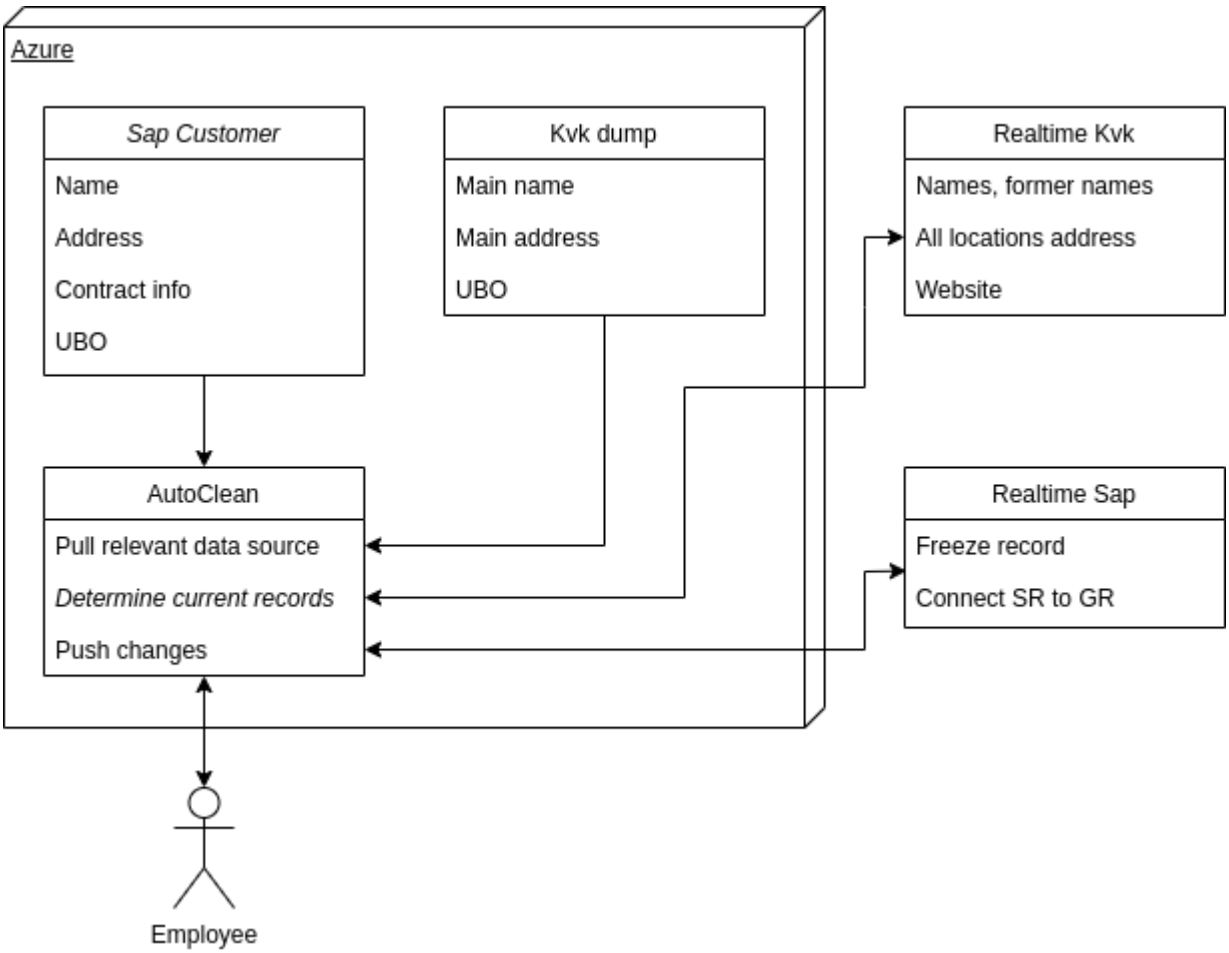


Figure 4: basic schematic overview of the future database design

5.1.2 Front-end

The front-end is a simple interface where the data objects passed from the back-end are displayed in a convenient way. In contrast to the old process, the values in a Golden Record are completely ignored as these values are a summary from the Silver Records (child) objects. As the HI might have made mistakes while creating this Golden Record, these values can be incorrect and are therefore not used.

The employee is guided by the Golden Record step by step. The first step is to check whether the HI software combined the Silver Records correctly; for this, the 'group' function can be used. Next, a legal entity from the Chamber of Commerce should be chosen. After searching in the database, the employee is presented with a list of possible matching Silver Records. The employee can click the button to add the Silver Record to the Golden Record.

Although the design of the front-end has been made for the purpose of this research, the implementation of the front-end was the responsibility of NN. Due to technical limitations, the program does not have write access to SAP BP, connecting a Silver Record to a Golden Record is done manually.

The initial design, as well as screenshots of the final web app, are in the appendix.

5.2 Automatic cleaning

The second step in improving the speed and accuracy of the cleaning process is to also takeover the human decision points in figure 2. As described by [15], the variables should be weighted dynamically. To achieve this a cleaning algorithm is implemented. While employees were using the web app, their decisions, and variables leading to these decisions were saved.

The variables used to determine if a Silver Record should be part of a Golden Record are extracted from table 2. Next to this, a name match score has been added; the full list of variables:

- Name match score: integer score between 0 and 100
- Address equal: boolean
- KvK equal: boolean
- Domain equal: boolean
- Phone number equal: boolean
- Connected by employee: boolean

The name match score is calculated through the Levenshtein distance, which does not take word order into account while calculating the score. The connected by employee value is used as target value.

During a two week period, 10 employees used the web app. This resulted in a dataset with 60.298 Silver Records of which 44.620 Silver Records have the status 'Finished' (Table 1); these values could be used to train the cleaning algorithm.

gr	sr	nameMatch	addresEqual	kvkEqual	domainEqual	phoneEqual	connecByEmp
000	999	96	1	0	0	0	1
001	998	46	0	0	0	0	0
002	997	83	1	1	0	0	1

Table 3: example of input table of algorithm

The input table is imported into RapidMiner [16]. After selecting the right target column and columns which can be ignored (sr, gr), based on the data RapidMiner runs several machine learning algorithms. In this case 8 independent algorithms where tested.

Model	Classification Error	Standard Deviation	Scoring time
Naive Bayes	0.9%	0.4%	86 ms
Generalized Linear Model	1.1%	0.4%	61 ms
Logistic Regression	0.9%	0.4%	56 ms
Fast Large Margin	1.0%	0.3%	50 ms
Deep Learning	1.0%	0.4%	115 ms
Decision Tree	1.1%	0.3%	68 ms
Random Forest	1.1%	0.3%	1 s
Gradient Boosted Trees	1.2%	0.4%	325 ms

Table 4: results of machine learning comparison

Since in a future state the accuracy should be as high as possible the Logistic Regression algorithm is chosen is it has the smallest execution time. Logistic Regression was 99.1% accurate in predicting if a Silver Record should be connected to a Golden Record. These result are measured by calculating the percentage of incorrect predictions. After manually inspecting the values which were not correctly classified, we saw and concluded that employees made manual errors while connecting a Silver Record to a Golden Record. After solving these issues, an accuracy of 99.6% could be obtained.

The model is created including all examples and generating a generic mathematical function. For example: if a Silver Record, with an 80% name match is connected by the employee and a Silver Record with a 60% name match is not. Based on Logistic Regression a record with a 71% match will be connected whereas a 69% match will not be connected. The accuracy is calculated while applying the mathematical function to data inputs. The result of the mathematical function is compared to the actual performance of the employee, counting all valid and invalid predictions of the algorithm. Dividing the two rates results in 99.6% accuracy compared the human employees.

To improve the accuracy of the ML over time, the algorithm will evolve from records looked at by an employee. This allows the algorithm to be able to automatically approve advanced records by itself and become 'self improving'.

6 Impact on productivity

The amount of solved records of every individual employee are recorded on a daily base. The error-rate is viewed weekly on a team base level. The Benefits of both the automated solution as the web app will be calculated from different viewpoints.

There are currently 270.271 not processed Golden Records. As Golden Records might be combined during the cleaning process, not all Golden Records should be looked at. Based on already cleaned records a Golden Records (63028) contain 8.81 Silver Records on Average. At the moment there are 1.396.418 not cleaned Silver Records. When extrapolating this, approximately 158.460 Golden Records need to be cleaned.

6.1 Time improvement

6.1.1 Web app

Two weeks of using the web app is compared to two weeks of not using the web app. Due to sickness and holiday the amount of days the web app is used is lower than the manual approach. As every employee was working from home, it is not possible to determine the time a employee was actually working.

Type	Days used	Mean	Standard Deviation
Manual	139	26,4	9,41
Web app	69	34,7	8,02

Table 4: Records cleaned per day per employee

Using a significant T-test and a confidence level of 99% there is significant proof the web app has a higher trough put time than the manual approach.

6.1.2 Automatic cleaning with Artificial Intelligence

Based on all 68.333 cleaned Golden Records:

- 1.70% has a 'Archived' status
- 13.53% has a 'BU assigned' status
- 8.47% has a 'Stopped' status
- 76.30% has a 'Finished' status

The 'Archived' and 'Stopped' Golden Records can be solved on a rule base and will be solved with a 100% accuracy. As the algorithm is not 100% accurate, the algorithm can only connect Silver Records to Golden Record if the accuracy is high. As we know the average Golden Record contains 8.81 Silver Records. There's a 99.6% chance of correctly identifying a Silver Record. Every Silver Record is provided with a percentage to display the certainty the algorithm is correct about determining the target value. For example, if the name, address and KvK values equal, the algorithm is 100% certain, the Silver Record should be connected to the Golden Record. Grouping false-positive Silver Records, a threshold of 80% accuracy is estimated. If all Silver Records which have been found meet this minimum threshold, the Golden Record status will be set to 'Finished' automatically; if this is not the case, an employee should take a look at this record. Based on the training set, 76% of the Golden Record can be cleaned without human interference. This group includes the 'BU assigned' group, as well as these, are more complicated records. If the Chamber of Commerce number can not be determined automatically, the Golden Record should be cleaned by an employee as well.

6.2 Costs savings

6.2.1 Web app

A cleaning employee is hired externally approximately for a €55,- hourly rate, there are 20 FTE. An estimation on the amount of days is calculated using table 4.

- Cleaning 158.460 records manually would take 300,11 working days, this will cost €2.640.968,-
- Cleaning 158.460 records in the web app would take 228.34 days, this will cost €2.009.291,-

Using the web app will result in finishing the cleaning process approximately 71.77 working days earlier, based on 20 full-time employees.

The investment costs of the web app are hard to predict, however this would be lower than the savings, as it took a inexperience trainee 6 months to built the entire tool. An API request to the Chamber of Commerce costs €0,016 per query. This would result in €2535.36. An external UI developer (hired by NN) has build the interface in two weeks. He is payed €85 per hour, this results

in a salary of €6800,-. If we assume a consultant architect need 3 months to inspect and build the back-end at a €110,- hourly rate, this would cost 110*8*5*52/4 = €57.200,-. Summed up, the total costs are estimated at €66.535,36. The cost savings are €565.141,64 which is a reduction of 21.4%.

6.2.2 Automatic cleaning with Artificial Intelligence

For the proposed Artificial Intelligence approach: based on historical data it is estimated that 1.7% + 8.47% = 10.17% of the Golden Records = 16.115 Golden Records can be processed by the rules. Of the remaining Golden Records, 89.83% * (1 - 0.76) = 21.56% can not be processed by the algorithm and should be processed by an employee (as calculated in section 6.1.2). This means 30.690 Golden Records should be processed via the web app, this will take 44.21 working days (based on 20 FTE), and will cost €389.147,-

Compared to the manual approach, 85.3% time is saved. If we assume an experienced developer / Artificial Intelligence expert needs an additional month to build the Artificial Intelligence into the web app at a €140 hourly rate, this would have costed approximately: 140*8*5*52/12 = €24.266,-. Bringing the total costs on €90.801,36 + €389.147,- = €479.948.36, this results in a cost reduction of 81.8% in comparison to the manual approach. Investing in Artificial Intelligence will therefore lead to a ROI of 450%.

7 Conclusions

In this section we look back on the project and look at the the advantages and disadvantages of the use of a web app and Artificial Intelligence approach. Furthermore we look at the options for future work. Below, the research questions will be covered, leading to the main research question.

- RQ.a: How can the quality of data cleaning by measured after the implementation of AI?
  The Artificial Intelligence has the same quality of cleaning above a certain threshold. Below this threshold, inconsistencies may occur and it is recommended to process these records via the web app. To improve the accuracy of the ML over time, the algorithm will evolve from records looked at by an employee. This allows the algorithm to be able to automatically approve advanced records by itself and become 'self improving'.

- RQ.b: Is it possible to automate steps from the process on a rule base?
  In the web app design, all manual steps are automated. Unfortunately the case infrastructure at NN still required some manual steps which delayed the process. Nevertheless, a speed increase of 31.4% compared to the old manual process has been found.

- RQ.c: Can the improvement after implementing AI be measured in practice?
  Golden Records which are cleaned by the Artificial Intelligence can be cleaned overnight and do not require human interference. Calculation shows that the estimated time improvement is 85.3%, this should be measured in practice to be confirmed. Due the higher rate of automation, the rate of human error is reduced further.

- RQ.d: What is the ROI of implementing AI in the data cleaning process?
  The true costs of implementing Artificial Intelligence in a business environment, as are the

time improvements are estimated. Based on these estimations a price calculation is made, this shows a ROI of 450%.

- RQ: To what extent can the data cleaning process be improved by implementing AI?
  To train the Artificial Intelligence algorithm and reduce repeated and manual tasks, a web app has been built which shows a significant impact on the amount of records cleaned. Artificial Intelligence has been implemented to cover most of the cases automatically, records below a threshold may include inconsistencies and should be covered in the web app by an employee. This results in an easier cleaning, decline in error rates and a significant reduction in time and money spend.

Other potentially interesting research would be to create a framework which automatically cleans comparable databases. Also, the Artificial Intelligence algorithm can be improved or (machine learning) algorithms can be used to further improve the quality of the automatically cleaned records. We also think implementing the Artificial Intelligence algorithm into the web app interface would speed up the process even more.

7.1 Recommendations

Based on this research, recommendations for NN are made:

- Automatically check records with rules, to cover 'Archived' and 'Stopped' companies. The remaining companies will be covered by Artificial Intelligence.
- Artificial Intelligence records below a threshold can contain inconsistencies and should be looked at by an employee.
- As the training set can discover outliers in the data, it is recommended to re-clean records cleaned with the old manual approach and check whether this set contains outliers.

References

[1] Raad van State, der Staten-Generaal (2008). *Wet ter voorkoming van witwassen en financieren van terrorisme.* https://wetten.overheid.nl/BWBR0024282/2020-07-10.

[2] Belastingdienst; Fiscale inlichtingen- en opsporingsdienst (2018). *ING betaalt 775 miljoen vanwege ernstige nalatigheden bij voorkomen witwassen.* https://www.fiod.nl/ing-betaalt-775-miljoen-vanwege-ernstige-nalatigheden-bij-voorkomen-witwassen.

[3] Object Management Group (2017). *OMG® Unified Modeling Language®.* https/www.omg.org/spec/UML/.

[4] R.J. Wieringa. (2014). *Design science methodology for information systems and software engineering.* Springer.

[5] Loshin, D. (2010). *Master Data Management.* Morgan Kaufmann.

[6] Haug, A. and Stentoft Arlbjørn, J. (2011). *Barriers to main data quality.* Journal of Enterprise Information Management, Vol. 24 No. 3, pp. 288-303.

[7] E. Rahm, H. Hai Do (2000). *Data Cleaning: Problems and Current Approaches.* University of Leipzig.

[8] U. Fayyad (1998). *Mining Database: Towards Algorithms for Knowledge Discovery.* IEEE Techn. Bulletin Data Engineering 21(1).

[9] L.M. Haas, R.J. Miller, B. Niswonger, M. Tork Roth, P.M. Schwarz, E.L. Wimmers (1999). *Transforming Heterogeneous Data with Database Middleware: Beyond Integration.* IEEE Data Engineering Bulletin, Vol 22 No. 1, pp. 31-36

[10] A.H. Doan, P. Domingos, A.Y. Levy (2000). *Learning Source Description for Data Integration.* Proc. 3rd Intl. Workshop The Web and Databases (WebDB)

[11] W.S. Li, S. Clifton (2000). *SEMINT: A Tool for Identifying Attribute Correspondences in Heterogeneous Databases Using Neural Networks.* Data and Knowledge Engineering, Vol 33 No. 1, pp. 49-84

[12] C. SAPia, G. Höfling, M. Müller, C. Hausdorf, H. Stoyan, U. Grimmer (1999). *On Supporting the Data Warehouse Design by Data Mining Techniques.* GI-Workshop Data Mining and Data Warehousing

[13] V. Wangikar, R. Deshmukh (2011). *Data Cleaning: Current Approachesand Issues.* Deptartment of Computer Science IT, Dr. Babasaheb Ambedkar Marathwada University, Aurangabad (MS), India

[14] N. Koudas, S. Sarawagi, D. Srivastava (2006). *Record Linkage: Similarity Measures and Algorithms.* International Conference on Management of Data, Chicago, Illinois, USA, June 27-29, 2006

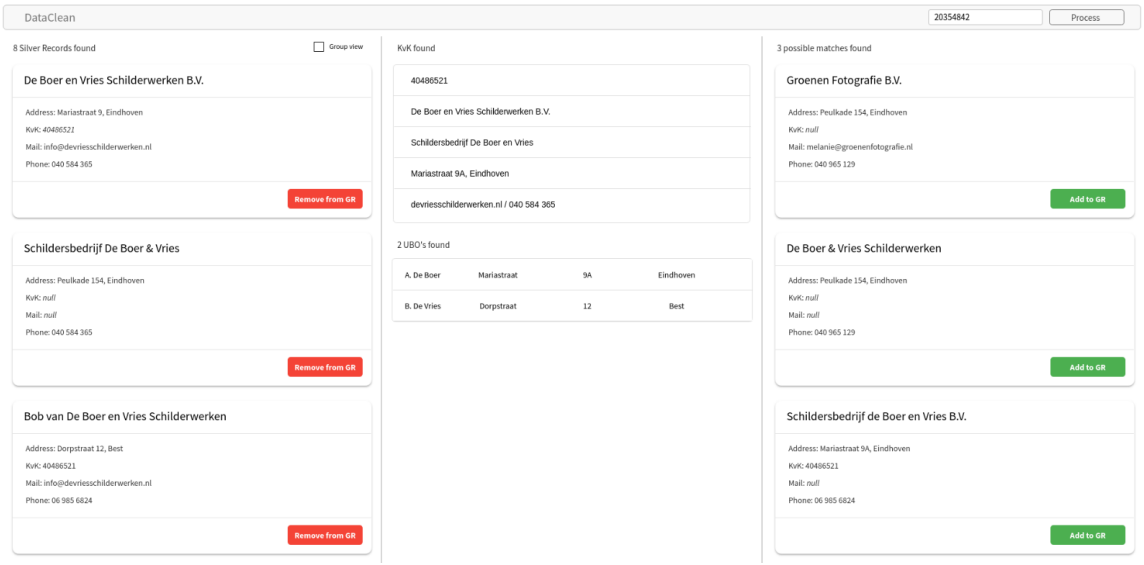
[15] S. Yan, D. Lee, M. Kany, C. Giles (2007). *Adaptive Sorted Neighborhood Methods for Efficient RecordLinkage.* ACM/IEEE Joint Conference on Digital Libraries, Vancouver, BC, Canada, June 18-23, 2007

[16] V. Kotu, B. Deshpande (2016). *Predictive Analytics and Data Mining: Concepts and Practice with RapidMiner.* Elsevier.

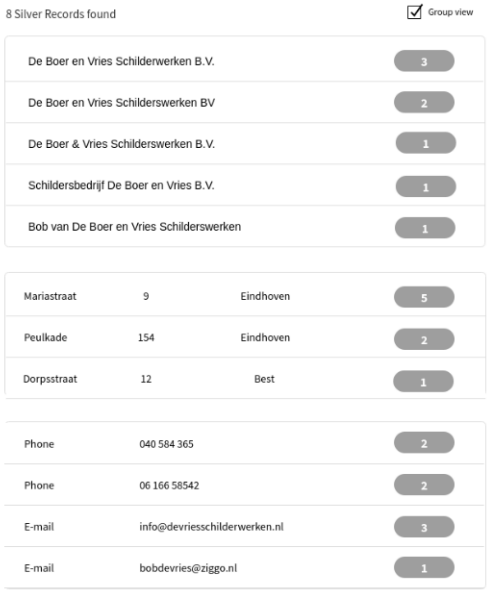
Appendix

UI design

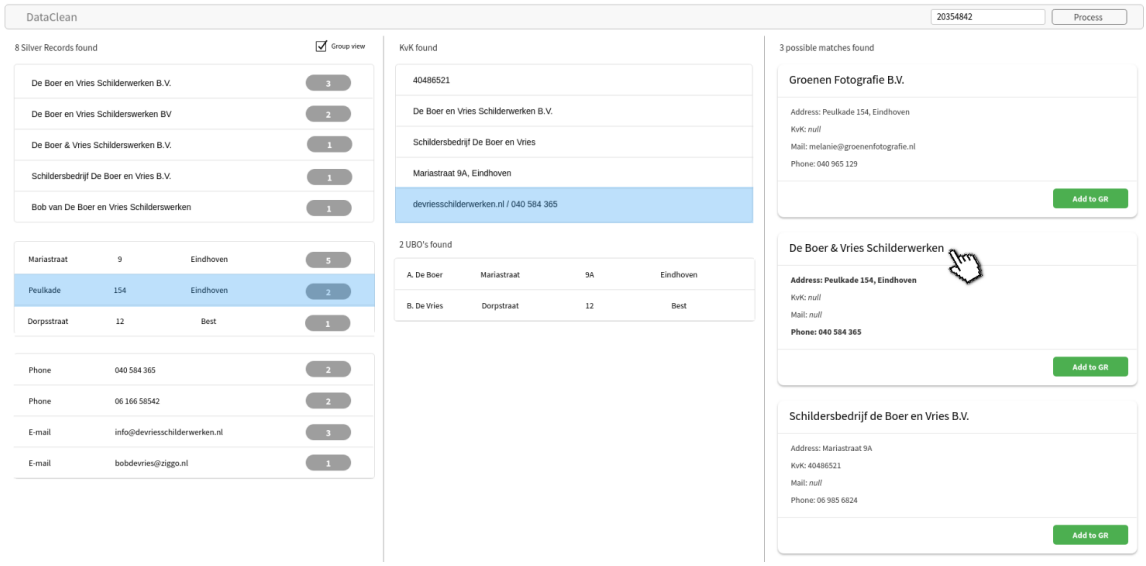
The initial design of the UI which the NN developer used to design the interface. In step 1, the user inserts the Golden Record number (which can't be obtained automatically in the NN case). After inserting, the relevant data is pulled from various sources. From left to right: Current Silver Records, KvK registration, UBO's and might related Silver Records. Multiple KvK records can be shown if there are several options.



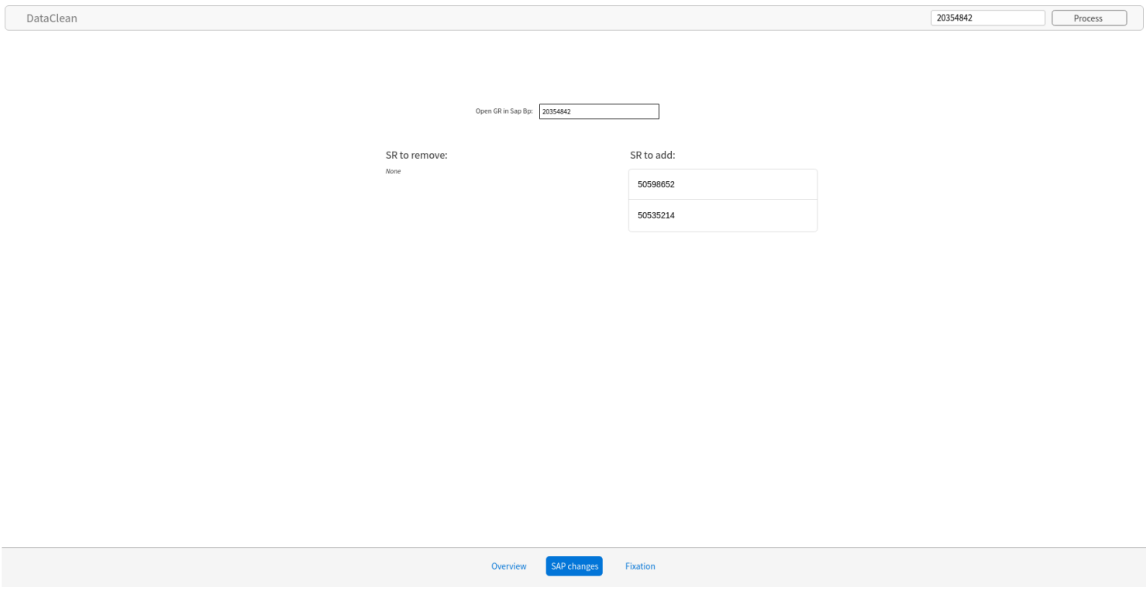
The 'Group Button' can be used by the employee to reveal corresponding values, which is increasingly valuable when a lot of Silver Records are present.



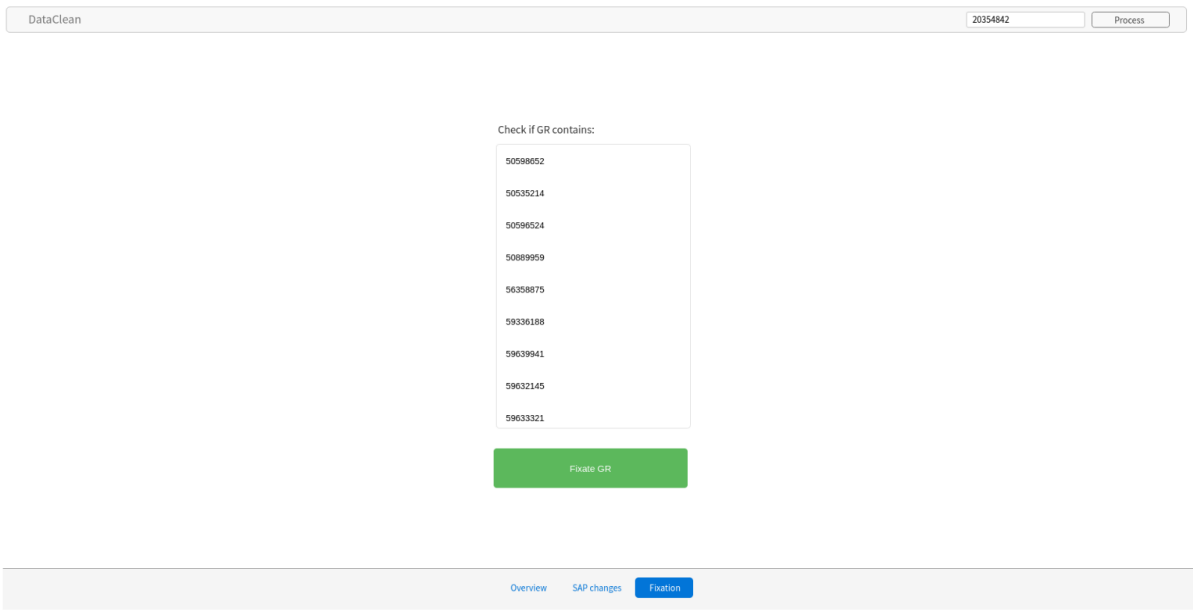
When hovering over a value, the matching values will be highlighted to the user:



On the bottom of the page three buttons are present which will lead to the next steps. In the 2th step the employee should manually make the changes in SAP BP, an overview screen tho make this process simpler is given:

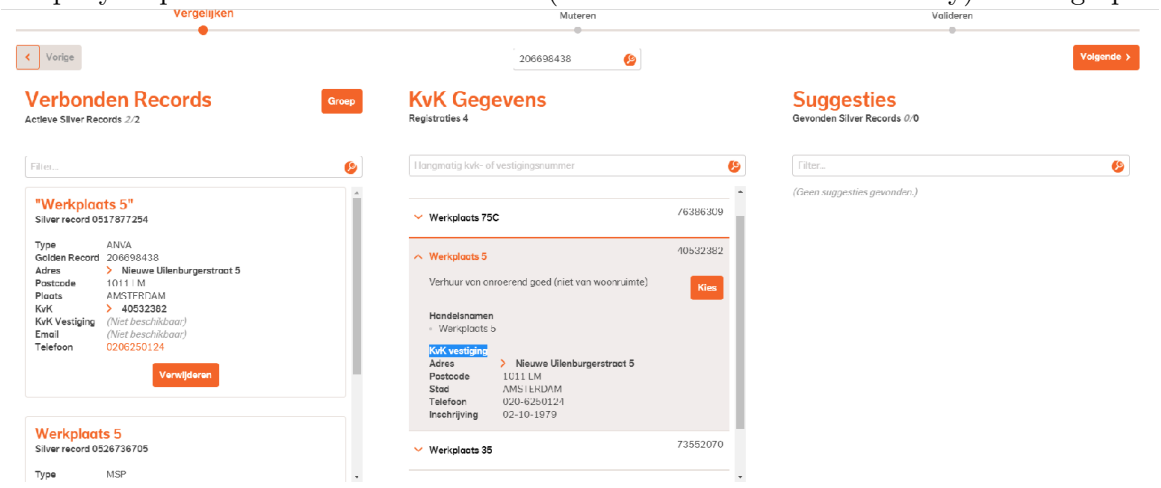


In the last step, an overview of the Silver Records that should be present in SAP BP is visible. This allows the employee to make a last check before proceeding to the next Golden Record.

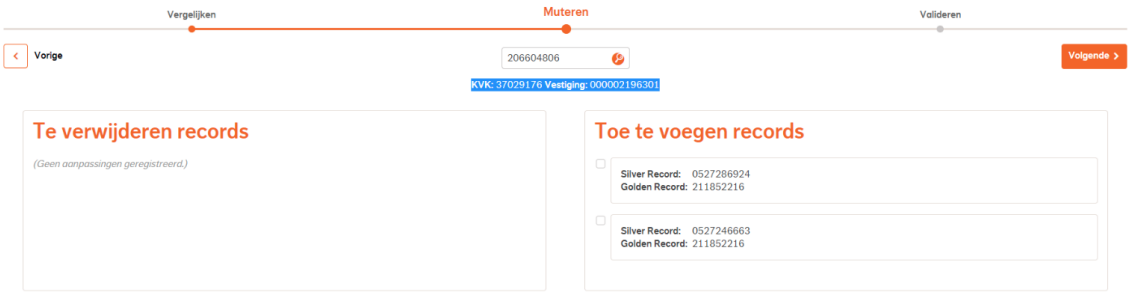


Final UI

After several sprints, meetings with the employees and data improvements the following UI has been delivered. Compared with the initial design the interface matches the cooperate identity of NN. Next to this, the company name, address and KvK number are clickable; leading to the KvK website with all details pre-filled. Also, a filter field has been implemented to search for addresses or company names. Finally, users can search on the department number of a company to prevent not relevant locations (on the other side of the country) showing up.



On the 'mutate' page, the KvK details are placed (click to copy) in order to speed up the process in SAP BP. The overview page has not changed compared to the design.



Back-end code

index.php

```
<?php
/*
This page (and all other files in this folder) is only loaded if
user is in  the NN network. This page will only redirecting to the UI
*/

header("Location: https://www.***.com/***");
die();
```

main.php

```
<?php
/*
The frond-end will post all the variables to this file. The supporting files
for accesing the right database tables are included below. A JSON with:
 - current Silver Records
 - KvK record
 - UBO details
 - possible Silver Record matches
will be returned to the user (and displayed in the UI)
*/

require __DIR__ . '/vendor/autoload.php';
include_once 'kvk.php';
include_once 'sapBp.php';
include_once 'azure.php';

use FuzzyWuzzy\Fuzz;
use FuzzyWuzzy\Process;

ini_set('display_errors', 1);
ini_set('display_startup_errors', 1);
error_reporting(E_ALL);

header('Content-Type: application/json');
header('Access-Control-Allow-Headers: *');
header('Access-Control-Allow-Origin: *');
header('Access-Control-Allow-Methods: GET,HEAD,OPTIONS,POST,PUT');
```

```
//validate Azure authentication token
$headers = apache_request_headers();
foreach ($headers as $header => $value) {
    if ($header == 'Authorization') {
        $token = base64_decode($value);
    }
}

$plain = base64_decode($token);
$plain = strstr($plain, "exp");
$unix = substr($plain, 5, 10);

$now = new Datetime();
if ($unix < $now->format('U')) {
    echo json_encode(array('error' => 'tokenExpired'));
    die();
}

//get lookup variables
$gr = $_GET['gr'];
$kvkNumber = '';
if (isset($_GET['kvk'])) {
    $kvkNumber = $_GET['kvk'];
}

getSrAndKvk($gr, $kvkNumber, $token);


function getSrAndKvk($grNumber, $kvkNumber, $token)
{
    $objectsArray = array();

    //goldenRecord number can't be correct
    if (empty($grNumber) || !is_numeric($grNumber) || strlen($grNumber) > 10) {
        echo json_encode(array('error' => 'grNotCorrect'));
        return false;
    }

    //get silverRecords
    $sr = sapGetChilds($grNumber, $token);
    array_push($objectsArray, $sr);

    //goldenRecord has no active products
    if (count($sr) === 0) {
        $kvk = getKvkNoSilverRecord($grNumber, $token);

        //if no kvk found, stop searching
        if (empty($kvk)) {
            echo json_encode(array('error' => 'noSuggedKvKNoActiveCustomer'));
            return false;
        }
    }

    $preventRecursive = false;
```

```
if (isset($kvk) || $kvkNumber !== '') {
    $preventRecursive = true;
}

//obtain most useful KvK record
if (!isset($kvk)) {
    $kvk = searchKvkRecord($kvkNumber, $sr, $preventRecursive, $token);

    if (empty($kvk)) {
        $kvkNumberArray = [];
        foreach ($sr as $record) {
            if (!empty($record['kvkNumber']) && $record['kvkNumber'] != 'NULL') {
                if (!in_array($record['kvkNumber'], $kvkNumberArray)) {
                    $kvkNumberArray[] = $record['kvkNumber'];
                }
            }
        }

        //silverRecords contains not unique kvk numbers
        if (count($kvkNumberArray) > 1) {
            $kvkNumber = '';
        }

        //if all KvK same
        if (count($kvkNumberArray) == 1) {
            $kvkNumber = $kvkNumberArray[0];
            $kvk = searchKvkRecord($kvkNumber, $sr, $preventRecursive, $token);
        }
    }
}

//if no KvK record has been found
if ($kvk === false) {
    array_push($objectsArray, array('error' => 'noSuggestedKvk'));
    echo json_encode($objectsArray);
    return false;
}

//multiple KvK search matches, ask employee right one
$kvkNumber = [];
foreach ($kvk as $record) {
    if (!empty($record['kvkNumber']) && $record['kvkNumber'] != 'NULL') {
        if (!in_array($record['kvkNumber'], $kvkNumber)) {
            $kvkNumber[] = $record['kvkNumber'];
        }
    }
}

if (count($kvkNumber) > 1) {
    array_push($objectsArray, $kvk);
    echo json_encode($objectsArray);
    return false;
}
```

```
    //KvK is right, get additional data from KvK API
    $kvkApi = @file_get_contents('https://api.kvk.nl/api/v2/profile/companies?
␣␣␣␣kvkNumber=' . $kvk[0]['kvkNumber'] . '&includeFormerTradeNames=
␣␣␣␣true&legalPerson=false&user_key=XXX');

    if (!$kvkApi) {//company has stopped recently, search for relaunch
        //kvk has been manually set and kvk is not active
        if ($preventRecursive === true) {
            array_push($objectsArray, array('error' => 'noSuggestedKvk'));
            echo json_encode($objectsArray);
            die();
        }

        $kvk = searchKvkRecord('', $sr, false, $token);
        array_push($objectsArray, $kvk);
    } else {//company active
        $kvkApi = json_decode($kvkApi, true);
        $kvkTradeNames = $kvkApi['data']['items'][0]['tradeNames'];

        $kvk[0]['companyNames'] = kvkActiveTradeNamesToString($kvkTradeNames);
        $kvk[0]['formerCompanyNames'] = kvkFormerTradeNamesToString($kvkTradeNames);

        array_push($objectsArray, $kvk);
    }

    //get UBO's
    if (count($sr) === 0) {
        $ubo = getUbo($grNumber, false, $token);
    } else {
        $ubo = getUbo($kvk[0]['kvkNumber'], true, $token);
    }

    array_push($objectsArray, $ubo);

    //search for missing silverRecords
    buildSqlMissingSilverRecords($sr, $kvk, $objectsArray, $ubo, $grNumber, $token);
}

function buildSqlMissingSilverRecords($sr, $kvk, $jsonArray, $ubo, $grNumber, $token)
{
    //KvK Number
    $query = '[kvkNumber]␣=␣\'' . $kvk[0]['kvkNumber'] . '\'';

    //address, combine current SR and kvk addresses
    $addressArray = [];
    $addresses = addressToArray($sr, $addressArray);
    $addresses = addressToArray($kvk, $addresses);

    //addresses
    foreach ($addresses as $address) {
        $address = explode(":", $address);

        //if NN's address has been filled in, ignore
        if ($address[0] == '1096␣BA' && $address[1] == '4') {
```

```
            continue;
        }

        if (!empty($address[0])) {//if zipCode has been filled in
            $query .= ' OR ([zipCode] = \'' . $address[0] . '\'
            AND [houseNumber] = \'' . $address[1] . '\')';
        } else if (!empty($address[4]) && !empty($address[5])) {
            $query .= ' OR ([streetName] = \'' . $address[4] . '\'
            AND [houseNumber] = \'' . $address[1] .
                '\' AND [city] = \'' . $address[1] . '\')';
        }
    }

    //trade names
    $tradeNames = tradeNamesNormalizeToArray($kvk, $sr);
    foreach ($tradeNames as $name) {
        $name = str_replace([' ', '-', "'", '.', '"'], '%', $name);
        $words = explode('%', $name);

        $skip = false;
        foreach ($words as $tradename_piece) {
            if (strlen($tradename_piece) < 4) {
                $skip = true;
            }
        }

        if ($skip) {
            continue;
        }

        if (sizeof($words) <= 1) {//too many results if trade name just 1 word
            $query .= ' OR [companyName] LIKE \'' . $name . '%\'';
            continue;
        }

        $query .= ' OR [companyName] LIKE \'';
        if ($name[0] = " ") {
            $query .= '###';
        }

        foreach ($words as $word) {
            for ($i = 0; $i < 3 && !empty($word[$i]); $i++) {
                $query .= $word[$i];
                if (empty($word[$i + 1]) || $i === 2) {
                    $query .= '%';
                    break;
                }
            }
        }
        $query .= '\'';
    }
    $query = str_replace(['###'], '%', $query);
    $query = preg_replace('/%+/', '%', $query);
```

```
//kvkPhoneNumber
$phoneNumbers = phoneNormalizeToArray($kvk, $ubo);
foreach ($phoneNumbers as $phoneNumber) {
    $phoneNumber = str_split($phoneNumber, 1);
    $phoneNumber = implode('%', $phoneNumber);
    $query .= '␣OR␣[phoneNumber]␣LIKE␣\'%' . $phoneNumber . '\'';
}

//kvkDomain
$websites = websiteToArray($kvk);
foreach ($websites as $website) {
    $query .= '␣OR␣[email]␣LIKE␣\'%' . $website . '\'';
}

//exclude already found SR (under current GR) from the results
if (count($sr) == 0) {
    $srNumbers = "";
} else {
    $srNumbers = srToString($sr);
}

$search = sapMasterSearch($query, $srNumbers, $grNumber, $token);

if (!empty($search)) {
    array_push($jsonArray, $search);
} else {
    $query = '';

    foreach ($tradeNames as $name) {
        $name = str_replace(['␣', '-', "'", '.', '"'], '%', $name);
        $words = explode('%', $name);

        $skip = false;
        foreach ($words as $tradename_piece) {
            if (strlen($tradename_piece) < 4) {
                $skip = true;
            }
        }

        if ($skip) {
            continue;
        }

        if (strlen($name) < 7) {
            continue;
        }

        if ($query !== '') {
            $query .= '␣OR';
        }

        if (sizeof($words) <= 1) {//too many results if trade name just 1 word
            $query .= '␣[companyName]␣LIKE␣\'%' . $name . '%\'';
            continue;
```

```
            }

            $query .= ' [companyName] LIKE \'%';

            foreach ($words as $word) {
                for ($i = 0; $i < 3 && !empty($word[$i]); $i++) {
                    $query .= $word[$i];
                    if (empty($word[$i + 1]) || $i === 2) {
                        $query .= '%';
                        break;
                    }
                }
            }
            $query .= '\'';
        }

        $query = preg_replace('/%+/', '%', $query);

        if (strlen($query) > 5) {
            $search = sapMasterSearch($query, $srNumbers, $grNumber, $token);
        }

        if (!empty($search)) {
            array_push($jsonArray, $search);
        } else {
            array_push($jsonArray, array());
        }
    }

    echo json_encode($jsonArray);

    if (count($search) > 0 && count($search) < 1000) {
        buildSqlMachineLearningTable($grNumber, $sr, $search, $kvk, $ubo, $token);
    }
}

function searchKvkRecord($kvkNumber, $sr, $manual, $token)
{
    //KvK can't be correct, search on address in KvK
    if (empty($kvkNumber) || !is_numeric($kvkNumber) || strlen($kvkNumber) !== 8) {
        //kvk department number is passed, search on this value
        if (is_numeric($kvkNumber) && strlen($kvkNumber) === 12) {
            return kvkGetDepartmentsByDepartment($kvkNumber, $token);
        }

        //search for department number in KvK
        $kvkDepartments = kvkDepartmentToString($sr);
        $relaunch = '';
        if (!empty($kvkDepartments)) {
            $relaunch = kvkCheckRelaunch($kvkDepartments, $token);

            if (count($relaunch) === 1) {
                $kvkNumber = $relaunch[0]['KVK'];
            }
```

```
        }

        if (empty($relaunch) || count($relaunch) > 1) {
            foreach ($sr as $record) {
                $zipCode = $record['zipCode'];
                $houseNumber = $record['houseNumber'];
                $streetName = str_replace("'", "", $record['streetName']);
                $city = str_replace("'", "", $record['city']);
            }

            //search on company name
            $tradeName = tradeNameNormalize($sr[0]['companyName']);
            $tradeName = str_replace(['"', "'", '␣'], '%', $tradeName);
            $pickKvk = kvkSearchName($tradeName, null, $token);

            //Search full address from silverRecord
            if (empty($pickKvk) && !empty($zipCode) && ($zipCode != '1096␣BA'
                && $houseNumber != '4')) {
                $pickKvk = kvkGetAddress($zipCode, $houseNumber,
                    $streetName, $city, $token);
            }

            //search just on company name
            if (empty($pickKvk)) {
                $pickKvk = kvkSearchName($tradeName, null, $token);
            }

            if (empty($pickKvk)) {
                return false;
            }

            return $pickKvk;
        }
    }

    //get all KvK locations
    $kvkDepartments = kvkGetDepartments($kvkNumber, $token);
    if (empty($kvkDepartments)) {//company stopped, search for re-launch
        if ($manual === true) {
            return false;
        }
        $kvkDepartments = searchKvkRecord('', $sr, false, $token);
    }

    return $kvkDepartments;
}

function buildSqlMachineLearningTable($grNumber, $sr, $suggestedSr, $kvk, $ubo, $token)
{
    //static fields for all suggested silverRecords
    $kvkTradeNames = tradeNamesNormalizeToArray($kvk, $sr);
    $kvkWebsites = websiteToArray($kvk);
    $kvkPhone = phoneNormalizeToArray($kvk, $ubo);
    $kvkNumber = $kvk[0]['kvkNumber'];
```

```
    $kvkGr = $kvk[0]['goldenRecord'];
    $kvkDepartment = kvkDepartmentToArray($kvk);

    foreach ($suggestedSr as $record) {
        array_push($sr, $record);
    }

    //list all official known addresses: KvK and UBO
    $kvkUbo = $kvk;
    foreach ($ubo as $uboRecord) {
        array_push($kvkUbo, $uboRecord);
    }

    //in order to prevent false positives
    if (count(checkFixed($grNumber, $token)) > 0) {
        return;
    }

    //calculate the variables for each current and found silverRecord
    $sqlInsert = 'DELETE [CC_DM_CAADQ].[DQ_SR_ML] WHERE [grNumber] = \'' . $grNumber .
    '\'; INSERT INTO [CC_DM_CAADQ].[DQ_SR_ML] (grNumber, grNumberKvk, srNumber,
    kvkNameMatchScore, kvkUboAddressMatch, kvkDomainMatch, kvkPhoneMatch, kvkEqual,
    datum) VALUES ';

    foreach ($sr as $silverRecord) {
        $kvkDomainMatch = 0;
        $kvkPhoneMatch = 0;
        $kvkEqual = 0;

        //kvkNameMatchMatch
        $kvkNameMatchMatch = fuzzCompareTradeName($silverRecord['companyName'],
            $kvkTradeNames);

        //kvkUboAddressMatch
        $kvkUboAddressMatch = compareAddressSrAndKvKUbo($silverRecord, $kvkUbo);

        //kvkDomainMatch
        if (!empty($silverRecord['email']) && !empty($kvkWebsites)) {
            $srDomain = emailGeneralize($silverRecord['email']);

            foreach ($kvkWebsites as $officialWebsite) {
                if ($officialWebsite == $srDomain) {
                    $kvkDomainMatch = 1;
                    break;
                }
            }
        }

        //kvkPhoneMatch
        if (!empty($silverRecord['phoneNumber']) && !empty($kvkPhone)) {
            $sapPhoneNumber = phoneNormalize($silverRecord['phoneNumber']);

            foreach ($kvkPhone as $officialPhone) {
                if ($officialPhone == $sapPhoneNumber) {
```

```
                    $kvkPhoneMatch = 1;
                    break;
                }
            }
        }

        //kvkEqual
        if (isset($silverRecord['kvkNumber'])
            && $silverRecord['kvkNumber'] == $kvkNumber) {
            $kvkEqual = 1;
        }

        //kvkDepartment equal
        if (isset($silverRecord['kvkNumber'])
            && array_key_exists($silverRecord['kvkDepartment'],
          $kvkDepartment)) {
            $kvkEqual = 1;
        }

        //add to SQL query
        $sqlInsert .= '(' . (int)$grNumber . ',' . (int)$kvkGr . ',' .
            int)$silverRecord['silverRecord'] . ',' . $kvkNameMatchMatch . ',' .
            $kvkUboAddressMatch . ',' . $kvkDomainMatch . ',' . $kvkPhoneMatch . ',' .
            $kvkEqual . ',\'' . date("Y-m-d_h:i:sa") . '\'),';
    }

    $sqlInsert = substr($sqlInsert, 0, -1);
    $sqlInsert .= ';';

    executeSQL($sqlInsert, $token);
}

function fuzzCompareTradeName($sapTradeName, $kvkTradeNames)
{
    $sapTradeName = tradeNameNormalize($sapTradeName);
    $fuzz = new Fuzz();
    $highestRatio = 0;

    foreach ($kvkTradeNames as $kvkTradeName) {
        $ratio = $fuzz->ratio($sapTradeName, $kvkTradeName);

        if ($ratio > $highestRatio) {
            if ($ratio === 100) {
                return 100;
            }

            $highestRatio = $ratio;
        }
    }

    return $highestRatio;
}

function compareAddressSrAndKvKUbo($silverRecord, $kvkUboAddress)
```

```
{
    $sapStreetName = $silverRecord['streetName'];
    $sapHouseNumber = $silverRecord['houseNumber'];
    $sapZipCode = $silverRecord['zipCode'];
    $sapCity = $silverRecord['city'];

    //check if the address of the SR matches official registered address
    foreach ($kvkUboAddress as $kvkAddress) {
        if (!empty($kvkAddress['zipCode']) && !empty($kvkAddress['houseNumber'])
            && $kvkAddress['zipCode'] == $sapZipCode
            && $kvkAddress['houseNumber'] == $sapHouseNumber) {
            return true;
        }

        if (!empty($kvkAddress['streetName']) && !empty($kvkAddress['houseNumber'])
            && $kvkAddress['streetName'] == $sapStreetName
            && $kvkAddress['houseNumber'] == $sapHouseNumber) {
            return true;
        }

        if (!empty($kvkAddress['city']) && !empty($sapCity)
            && $kvkAddress['city'] == $sapCity
            && empty($sapStreetName) && empty($sapZipCode)) {
            return true;
        }

        if (!empty($kvkAddress['postZipCode'])
            && !empty($kvkAddress['postNumber'])
            && !empty($kvkAddress['postCity'])
            && $kvkAddress['postZipCode'] == $sapZipCode
            && $kvkAddress['postNumber'] == $sapHouseNumber
            && $kvkAddress['postCity'] == $sapCity) {
            return true;
        }
    }

    return 0;
}

function srToString($sr)
{
    $srNumbersArray = [];
    $srNumbers = '';
    foreach ($sr as $record) {
        if (!in_array($record['silverRecord'], $srNumbersArray)) {
            $srNumbersArray[] = $record['silverRecord'];
            $srNumbers .= '\'' . $record['silverRecord'] . '\',';
        }
    }
    $srNumbers = rtrim($srNumbers, ",");//remove last comma

    return $srNumbers;
}
```

```
function kvkDepartmentToString($sr)
{
    $kvkDepartmentArray = [];
    $kvkDepartments = '';
    foreach ($sr as $record) {
        if (!in_array($record['kvkDepartment'], $kvkDepartmentArray)) {
            $srNumbersArray[] = $record['kvkDepartment'];
            $kvkDepartments .= '\'' . $record['kvkDepartment'] . '\'' . ',';
        }
    }
    $kvkDepartments = rtrim($kvkDepartments, ",");//remove last comma

    return $kvkDepartments;
}

function kvkDepartmentToArray($sr)
{
    $kvkDepartmentArray = [];
    foreach ($sr as $record) {
        if (!in_array($record['kvkDepartment'], $kvkDepartmentArray)) {
            $srNumbersArray[] = $record['kvkDepartment'];
        }
    }

    return $kvkDepartmentArray;
}


function addressToArray($records, $addressArray)
{
    foreach ($records as $record) {
        $addressArray = addressToArrayHelper($record, $addressArray);
    }

    return $addressArray;
}

function addressToArrayHelper($record, $addressArray)
{
    $address = $record['zipCode'] . ':' . $record['houseNumber'] . ':' .
        houseNumberAdditionNormalize($record['houseNumberAddition']) . ':'
        . $record['streetName'] . ':' . $record['city'];
    $address2 = ':' . $record['houseNumber'] . ':' .
        houseNumberAdditionNormalize($record['houseNumberAddition'])
        . ':' . $record['streetName'] . ':' . $record['city'];

    if (!in_array($address, $addressArray)
        && !in_array($address2, $addressArray)) {
        $addressArray[] = $address;
    }

    return $addressArray;
}
```

```
function tradeNamesNormalizeToArray($kvk, $sr)
{
    $tradeNameArray = [];
    foreach ($kvk as $department) {
        $tradeNames = $department['companyNames'];
        $tradeNames = explode('||', $tradeNames);

        foreach ($tradeNames as $tradeName) {
            if (empty($tradeName) || $tradeName == '␣') {
                continue;
            }

            $generalizedTradeName = tradeNameNormalize($tradeName);
            if (!in_array($generalizedTradeName, $tradeNameArray)
                && $generalizedTradeName != '') {
                $tradeNameArray[] = $generalizedTradeName;
            }
        }

        if (isset($department['formerCompanyNames'])) {
            $tradeNames = $department['formerCompanyNames'];
            $tradeNames = explode('||', $tradeNames);

            foreach ($tradeNames as $tradeName) {
                if (empty($tradeName) || $tradeName == '␣') {
                    continue;
                }

                $generalizedTradeName = tradeNameNormalize($tradeName);
                if (!in_array($generalizedTradeName, $tradeNameArray)
                    && $generalizedTradeName != '') {
                    $tradeNameArray[] = $generalizedTradeName;
                }
            }
        }
    }

    if (isset($sr)) {
        foreach ($sr as $department) {
            $tradeName = $department['companyName'];

            if (empty($tradeName) || $tradeName == '␣') {
                continue;
            }

            $generalizedTradeName = tradeNameNormalize($tradeName);
            if (!in_array($generalizedTradeName, $tradeNameArray)
                && $generalizedTradeName != '') {
                $tradeNameArray[] = $generalizedTradeName;
            }
        }
    }

    return $tradeNameArray;
```

```
}

function kvkTradeNamesHelper($tradeNames, $tradeNamesArray, $string)
{
    if ($string) {
        if (!in_array($tradeNames, $tradeNamesArray) && $tradeNames != '') {
            $tradeNamesArray[] = $tradeNames;
        }

        return $tradeNamesArray;
    }

    foreach ($tradeNames as $tradeName) {
        if (empty($tradeName) || $tradeName == '␣') {
            continue;
        }

        if (!in_array($tradeName, $tradeNamesArray) && $tradeName != '') {
            $tradeNamesArray[] = $tradeName;
        }
    }
    return $tradeNamesArray;
}

function kvkActiveTradeNamesToString($kvkTradeNames)
{
    $tradeNamesArray = [];

    if (!empty($kvkTradeNames['currentStatutoryNames'])) {
        $tradeNamesArray = kvkTradeNamesHelper($kvkTradeNames['currentStatutoryNames'],
            $tradeNamesArray, false);
    }

    if (!empty($kvkTradeNames['currentTradeNames'])) {
        $tradeNamesArray = kvkTradeNamesHelper($kvkTradeNames['currentTradeNames'],
            $tradeNamesArray, false);
    }

    if (!empty($kvkTradeNames['businessName'])) {
        $tradeNamesArray = kvkTradeNamesHelper($kvkTradeNames['businessName'],
            $tradeNamesArray, true);
    }

    if (!empty($kvkTradeNames['shortBusinessName'])) {
        $tradeNamesArray = kvkTradeNamesHelper($kvkTradeNames['shortBusinessName'],
            $tradeNamesArray, true);
    }

    $tradeNameString = '';
    foreach ($tradeNamesArray as $tradeName) {
        $tradeNameString .= '||' . $tradeName;
    }

    return substr($tradeNameString, 2);
```

```
}

function kvkFormerTradeNamesToString($kvkTradeNames)
{
    $tradeNamesArray = [];

    if (!empty($kvkTradeNames['formerStatutoryNames'])) {
        $tradeNamesArray = kvkTradeNamesHelper($kvkTradeNames['formerStatutoryNames'],
            $tradeNamesArray, false);
    }

    if (!empty($kvkTradeNames['formerTradeNames'])) {
        $tradeNamesArray = kvkTradeNamesHelper($kvkTradeNames['formerTradeNames'],
            $tradeNamesArray, false);
    }

    $tradeNameString = '';
    foreach ($tradeNamesArray as $tradeName) {
        $tradeNameString .= '||' . $tradeName;
    }

    return substr($tradeNameString, 2);
}

function phoneNormalizeToArray($kvk, $ubo)
{
    $phoneNumberArray = [];

    //kvk phone Number
    foreach ($kvk as $department) {
        $phoneNumber = $department['phoneNumber'];
        if (empty($phoneNumber) || $phoneNumber == '␣') {
            continue;
        }

        $phoneNumber = phoneNormalize($phoneNumber);
        if (!in_array($phoneNumber, $phoneNumberArray)) {
            $phoneNumberArray[] = $phoneNumber;
        }
    }

    //landline UBO
    foreach ($ubo as $person) {
        $phoneNumber = $person['phone'];
        if (empty($phoneNumber) || $phoneNumber == '␣') {
            continue;
        }

        $phoneNumber = phoneNormalize($phoneNumber);
        if (!in_array($phoneNumber, $phoneNumberArray)) {
            $phoneNumberArray[] = $phoneNumber;
        }
    }
```

```
    //mobile phone UBO
    foreach ($ubo as $person) {
        $phoneNumber = $person['mobilePhone'];
        if (empty($phoneNumber) || $phoneNumber == '␣') {
            continue;
        }

        $phoneNumber = phoneNormalize($phoneNumber);
        if (!in_array($phoneNumber, $phoneNumberArray)) {
            $phoneNumberArray[] = $phoneNumber;
        }
    }

    return $phoneNumberArray;
}

function websiteToArray($kvk)
{
    $websiteArray = [];
    foreach ($kvk as $department) {
        $website = $department['website'];
        if (empty($website) || $website == '␣') {
            continue;
        }

        $website = domainNormalize($website);
        if (!in_array($website, $websiteArray)) {
            $websiteArray[] = $website;
        }
    }

    return $websiteArray;
}

function tradeNameNormalize($tradeName)
{
    $toSplit = array('␣ho␣', '␣h.o.␣', '␣HO␣', '␣hodn␣', '␣h.o.d.n␣', '␣inzake␣',
        '␣tav␣', '␣t.a.v.␣','␣tbv␣', '␣t.b.v.␣');
    $split = str_replace($toSplit, '#####', $tradeName);
    $split = explode('#####', $split);

    if (isset($split[0]) && count($split) > 1) {
        $tradeName = $split[1];
    }

    $toRemove = array('"', '-', 'B.V.', 'b.v.', 'bv', 'BV', 'Bv,', 'VOF', 'vof',
        'V.O.F.', 'v.o.f.', 'Vof','␣NV', 'N.V.', '␣nv', '␣n.v.', 'st␣', 'stichting␣',
        'st.', 'St␣', 'St.', '␣en', '␣&', 'zzp', 'z.z.p.', 'ZZP', 'Z.Z.P.', 'CV',
        'cv␣', 'C.V.', 'c.v.', 'mts', 'mts.', 'Mts', 'Mts.', '␣i.o.', '␣I.O.', 'RK',
        'parochie', 'Parochie', '␣te', 'Bestuur␣van', 'Best␣', 'Kerkvoogdij', 'Bw␣',
        'bw␣', 'Hervormde', 'hervormde', 'VvE', 'vve', 'VVE', 'V.v.E', 'V.v.E',
        'Vereniging␣van␣Eigenaars', 'Vereniging␣van␣eigenaars');
    $tradeName = str_replace($toRemove, '', $tradeName);
    $tradeName = str_replace('␣␣', '␣', $tradeName);
```

```
    if (substr($tradeName, -1) == '␣') {
        $tradeName = substr($tradeName, 0, -1);
    }

    return $tradeName;
}

function houseNumberAdditionNormalize($houseNumberAddition)
{
    $houseNumberAddition = preg_replace('/[^a-zA-Z0-9]/', '', $houseNumberAddition);
    $houseNumberAddition = strtolower($houseNumberAddition);

    $toRemove = array('␣', 'bus');
    $houseNumberAddition = str_replace($toRemove, '', $houseNumberAddition);

    $houseNumberAddition = strtoupper($houseNumberAddition);
    return $houseNumberAddition;
}

function phoneNormalize($phoneNumber)
{
    $phoneNumber = preg_replace("/[^0-9]/", '', $phoneNumber);

    $toRemove = array('31', '0031', '3100');
    $phoneNumber = str_replace($toRemove, '', $phoneNumber);

    if ($phoneNumber[0] == 0) {
        $phoneNumber = substr($phoneNumber, 1);
    }

    return $phoneNumber;
}

function emailGeneralize($email)
{
    $email = strtolower($email);
    $email = explode('@', $email);

    return $email[1];
}

function domainNormalize($domain)
{
    $toRemove = array('www.', 'http', 'https', '://', '␣');
    return str_replace($toRemove, '', $domain);
}
```

kvk.php

```
<?php
/*
All KvK related queries to the NN KVK dump will be pulled from here.
Realtime KvK API is called in main.php
*/
```

```
function kvkGetDepartments($kvkNumber, $token)
{
    $result= executeSQL('
        SELECT
            LEFT(kvk.[Kamer_van_Koophandel_nummer], 8) as kvkNumber,
            kd.[REL_NR_GOUD] as goldenRecord,
            kvk.[Vestigingsnummer_KVK] as kvkDepartment,
            kvk.[Bedrijfsnaam] + \'||\' + kvk.[handelsnaam1] + \'||\'
                + kvk.[handelsnaam2] + \'||\'
                + kvk.[handelsnaam3] as companyNames,
            kvk.[Straatnaam] as streetName,
            kvk.[Huisnummer] as houseNumber,
            kvk.[Huisnummer_toevoeging] as houseNumberAddition,
            kvk.[Postcode] as zipCode,
            kvk.[Woonplaats] as city,
            kvk.[Postbusnummer] as postNumber,
            kvk.[Postcode_Postbusnummer] as postZipCode,
            kvk.[Vestigingsplaats_Postbusnummer] as postCity,
            kvk.[Telefoonnummer] as phoneNumber,
            stuff(kvk.[Webadres], 1, 4, \'\') as website,
            kvk.[Branchecode_omschrijving] as activityDescription,
            kvk.[Datum_oprichting] as incorporationDate
        FROM [CC_DM_CAADQ].[KVK] as kvk
        LEFT OUTER JOIN [CC_DM_CAADQ].[KLANT_DETAIL] as kd
            ON (LEFT(kvk.[Kamer_van_Koophandel_nummer], 8) = kd.[KVK_NR]
            AND kvk.[Vestigingsnummer_KVK] = kd.[VESTIGINGSNUMMER])
        WHERE [Kamer_van_Koophandel_nummer] LIKE \'' . $kvkNumber . '%\'
        AND kd.[REL_NR_GOUD] IN (
            SELECT DISTINCT([BEDRIJF_GOUD])
            FROM [CC_DM_CAADQ].[UBO_LIJST]
            WHERE [BEDRIJF_AUGRP] = \'Z036\'
            AND [VESTIGING] = kvk.[Vestigingsnummer_KVK]
        )
        ORDER BY kvk.[Kamer_van_Koophandel_nummer]
    ', $token);

    if (count($result) > 0) {
        return $result;
    }

    return executeSQL('
        SELECT
            LEFT(kvk.[Kamer_van_Koophandel_nummer], 8) as kvkNumber,
            kd.[REL_NR_GOUD] as goldenRecord,
            kvk.[Vestigingsnummer_KVK] as kvkDepartment,
            kvk.[Bedrijfsnaam] + \'||\' + kvk.[handelsnaam1] + \'||\'
                + kvk.[handelsnaam2] + \'||\'
                + kvk.[handelsnaam3] as companyNames,
            kvk.[Straatnaam] as streetName,
            kvk.[Huisnummer] as houseNumber,
            kvk.[Huisnummer_toevoeging] as houseNumberAddition,
            kvk.[Postcode] as zipCode,
            kvk.[Woonplaats] as city,
```

```
            kvk.[Postbusnummer] as postNumber,
            kvk.[Postcode_Postbusnummer] as postZipCode,
            kvk.[Vestigingsplaats_Postbusnummer] as postCity,
            kvk.[Telefoonnummer] as phoneNumber,
            stuff(kvk.[Webadres], 1, 4, \'\') as website,
            kvk.[Branchecode_omschrijving] as activityDescription,
            kvk.[Datum_oprichting] as incorporationDate
        FROM [CC_DM_CAADQ].[KVK] as kvk
        LEFT OUTER JOIN [CC_DM_CAADQ].[KLANT_DETAIL] as kd
            ON LEFT(kvk.[Kamer_van_Koophandel_nummer], 8) = kd.[KVK_NR]
        WHERE [Kamer_van_Koophandel_nummer] LIKE \'' . $kvkNumber . '%\'
        AND kd.[REL_NR_GOUD] IN (
            SELECT DISTINCT(RIGHT([BEDRIJF_GOUD],9))
            FROM [CC_DM_CAADQ].[UBO_LIJST]
            WHERE [BEDRIJF_AUGRP] = \'Z036\'
            AND [KVK] = \'' . $kvkNumber . '\'
        )
        ORDER BY kvk.[Kamer_van_Koophandel_nummer]
    ', $token);
}

function kvkGetDepartmentsByDepartment($kvkDepartment, $token)
{
    return executeSQL('
        SELECT
            LEFT(kvk.[Kamer_van_Koophandel_nummer], 8) as kvkNumber,
            kd.[REL_NR_GOUD] as goldenRecord,
            kvk.[Vestigingsnummer_KVK] as kvkDepartment,
            kvk.[Bedrijfsnaam] + \' || \' + kvk.[handelsnaam1] + \' || \'
                + kvk.[handelsnaam2] + \' || \'
                + kvk.[handelsnaam3] as companyNames,
            kvk.[Straatnaam] as streetName,
            kvk.[Huisnummer] as houseNumber,
            kvk.[Huisnummer_toevoeging] as houseNumberAddition,
            kvk.[Postcode] as zipCode,
            kvk.[Woonplaats] as city,
            kvk.[Telefoonnummer] as phoneNumber,
            stuff(kvk.[Webadres], 1, 4, \'\') as website,
            kvk.[Branchecode_omschrijving] as activityDescription,
            kvk.[Datum_oprichting] as incorporationDate
        FROM [CC_DM_CAADQ].[KVK] as kvk
        LEFT OUTER JOIN [CC_DM_CAADQ].[KLANT_DETAIL] as kd
            ON (LEFT(kvk.[Kamer_van_Koophandel_nummer], 8) = kd.[KVK_NR]
            AND kvk.[Vestigingsnummer_KVK] = kd.[VESTIGINGSNUMMER])
        WHERE kvk.[Vestigingsnummer_KVK] = \'' . $kvkDepartment . '\'
        AND kd.[REL_NR_GOUD] IN (
            SELECT DISTINCT([BEDRIJF_GOUD])
            FROM [CC_DM_CAADQ].[UBO_LIJST]
            WHERE [BEDRIJF_AUGRP] = \'Z036\'
            AND [VESTIGING] = kvk.[Vestigingsnummer_KVK]
        )
        ORDER BY kvk.[Kamer_van_Koophandel_nummer]
    ', $token);
}
```

```
function kvkGetAddress($zipcode, $housenumber, $streetName, $city, $token)
{
    return executeSQL('
        SELECT
            DISTINCT LEFT(kvk.[Kamer_van_Koophandel_nummer], 8) as kvkNumber,
            kd.[REL_NR_GOUD] as goldenRecord,
            kvk.[Vestigingsnummer_KVK] as kvkDepartment,
            kvk.[Bedrijfsnaam] + \'||\' + kvk.[handelsnaam1] + \'||\'
                + kvk.[handelsnaam2] + \'||\'
                + kvk.[handelsnaam3] as companyNames,
            kvk.[Straatnaam] as streetName,
            kvk.[Huisnummer] as houseNumber,
            kvk.[Huisnummer_toevoeging] as houseNumberAddition,
            kvk.[Postcode] as zipCode,
            kvk.[Woonplaats] as city,
            kvk.[Telefoonnummer] as phoneNumber,
            stuff(kvk.[Webadres], 1, 4, \'\') as website,
            kvk.[Branchecode_omschrijving] as activityDescription,
            kvk.[Datum_oprichting] as incorporationDate
        FROM [CC_DM_CAADQ].[KVK] as kvk
        LEFT OUTER JOIN [CC_DM_CAADQ].[KLANT_DETAIL] as kd
            ON (LEFT(kvk.[Kamer_van_Koophandel_nummer], 8) = kd.[KVK_NR]
            AND kvk.[Vestigingsnummer_KVK] = kd.[VESTIGINGSNUMMER])
        WHERE kvk.[Kamer_van_Koophandel_nummer] != \' \'
        AND kvk.[Kamer_van_Koophandel_nummer] IS NOT NULL
        AND (kvk.[Postcode] = \'' . $zipcode . '\'
            AND kvk.[Huisnummer] = \'' . $housenumber . '\')
        OR (kvk.[Straatnaam] = \'' . $streetName . '\'
            AND kvk.[Huisnummer] = \'' . $housenumber . '\'
            AND kvk.[Woonplaats] = \'' . $city . '\')
        AND kd.[REL_NR_GOUD] IN (
            SELECT DISTINCT([BEDRIJF_GOUD])
            FROM [CC_DM_CAADQ].[UBO_LIJST]
            WHERE [BEDRIJF_AUGRP] = \'Z036\'
            AND [VESTIGING] = kvk.[Vestigingsnummer_KVK]
        )
    ', $token);
}

function kvkSearchName($tradeName, $city, $token)
{
    if ($city === null) {
        $result = executeSQL('
            SELECT
                LEFT(kvk.[Kamer_van_Koophandel_nummer], 8) as kvkNumber,
                kd.[REL_NR_GOUD] as goldenRecord,
                kvk.[Vestigingsnummer_KVK] as kvkDepartment,
                kvk.[Bedrijfsnaam] + \'||\' + kvk.[handelsnaam1] + \'||\'
                    + kvk.[handelsnaam2] + \'||\'
                    + kvk.[handelsnaam3] as companyNames,
                kvk.[Straatnaam] as streetName,
                kvk.[Huisnummer] as houseNumber,
                kvk.[Huisnummer_toevoeging] as houseNumberAddition,
```

```
                kvk.[Postcode] as zipCode,
                kvk.[Woonplaats] as city,
                kvk.[Telefoonnummer] as phoneNumber,
                stuff(kvk.[Webadres], 1, 4, '') as website,
                kvk.[Branchecode_omschrijving] as activityDescription,
                kvk.[Datum_oprichting] as incorporationDate
            FROM [CC_DM_CAADQ].[KVK] as kvk
            LEFT OUTER JOIN [CC_DM_CAADQ].[KLANT_DETAIL] as kd
                ON (LEFT(kvk.[Kamer_van_Koophandel_nummer], 8) = kd.[KVK_NR]
                AND kvk.[Vestigingsnummer_KVK] = kd.[VESTIGINGSNUMMER])
            WHERE kvk.[Kamer_van_Koophandel_nummer] != ''
            AND kvk.[Kamer_van_Koophandel_nummer] IS NOT NULL
            AND kvk.[Bedrijfsnaam] LIKE '%' . $tradeName . '%'
            AND kd.[REL_NR_GOUD] IN (
                SELECT DISTINCT([BEDRIJF_GOUD])
                FROM [CC_DM_CAADQ].[UBO_LIJST]
                WHERE [BEDRIJF_AUGRP] = 'Z036'
                AND [VESTIGING] = kvk.[Vestigingsnummer_KVK]
            )', $token);

        if(count($result) > 0) {
            return $result;
        }

        return executeSQL('
            SELECT
                LEFT(kvk.[Kamer_van_Koophandel_nummer], 8) as kvkNumber,
                kd.[REL_NR_GOUD] as goldenRecord,
                kvk.[Vestigingsnummer_KVK] as kvkDepartment,
                kvk.[Bedrijfsnaam] + '||' + kvk.[handelsnaam1] + '||'
                    + kvk.[handelsnaam2] + '||'
                    + kvk.[handelsnaam3] as companyNames,
                kvk.[Straatnaam] as streetName,
                kvk.[Huisnummer] as houseNumber,
                kvk.[Huisnummer_toevoeging] as houseNumberAddition,
                kvk.[Postcode] as zipCode,
                kvk.[Woonplaats] as city,
                kvk.[Telefoonnummer] as phoneNumber,
                stuff(kvk.[Webadres], 1, 4, '') as website,
                kvk.[Branchecode_omschrijving] as activityDescription,
                kvk.[Datum_oprichting] as incorporationDate
            FROM [CC_DM_CAADQ].[KVK] as kvk
            LEFT OUTER JOIN [CC_DM_CAADQ].[KLANT_DETAIL] as kd
                ON LEFT(kvk.[Kamer_van_Koophandel_nummer], 8) = kd.[KVK_NR]
            WHERE kvk.[Kamer_van_Koophandel_nummer] != ''
            AND kvk.[Kamer_van_Koophandel_nummer] IS NOT NULL
            AND kvk.[Bedrijfsnaam] LIKE '%' . $tradeName . '%'
            AND kd.[REL_NR_GOUD] IN (
                SELECT DISTINCT(RIGHT([BEDRIJF_GOUD],9))
                FROM [CC_DM_CAADQ].[UBO_LIJST]
                WHERE [BEDRIJF_AUGRP] = 'Z036'
                AND [KVK] = LEFT(kvk.[Kamer_van_Koophandel_nummer], 8)
            )', $token);
    }
```

```
    return executeSQL('
        SELECT
            LEFT(kvk.[Kamer_van_Koophandel_nummer], 8) as kvkNumber,
            kd.[REL_NR_GOUD] as goldenRecord,
            kvk.[Vestigingsnummer_KVK] as kvkDepartment,
            kvk.[Bedrijfsnaam] + \'||\' + kvk.[handelsnaam1] + \'||\'
                + kvk.[handelsnaam2] + \'||\'
                + kvk.[handelsnaam3] as companyNames,
            kvk.[Straatnaam] as streetName,
            kvk.[Huisnummer] as houseNumber,
            kvk.[Huisnummer_toevoeging] as houseNumberAddition,
            kvk.[Postcode] as zipCode,
            kvk.[Woonplaats] as city,
            kvk.[Telefoonnummer] as phoneNumber,
            stuff(kvk.[Webadres], 1, 4, \'\') as website,
            kvk.[Branchecode_omschrijving] as activityDescription,
            kvk.[Datum_oprichting] as incorporationDate
        FROM [CC_DM_CAADQ].[KVK] as kvk
        LEFT OUTER JOIN [CC_DM_CAADQ].[KLANT_DETAIL] as kd
            ON (LEFT(kvk.[Kamer_van_Koophandel_nummer], 8) = kd.[KVK_NR]
            AND kvk.[Vestigingsnummer_KVK] = kd.[VESTIGINGSNUMMER])
        WHERE kvk.[Kamer_van_Koophandel_nummer] != \' \'
        AND kvk.[Kamer_van_Koophandel_nummer] IS NOT NULL
        AND kvk.[Bedrijfsnaam] LIKE \'%' . $tradeName . '%\'
        AND kvk.[Woonplaats] = \'' . $city . '\'
        AND kd.[REL_NR_GOUD] IN (
            SELECT DISTINCT([BEDRIJF_GOUD])
            FROM [CC_DM_CAADQ].[UBO_LIJST]
            WHERE [BEDRIJF_AUGRP] = \'Z036\'
            AND [VESTIGING] = kvk.[Vestigingsnummer_KVK]
        )
    ', $token);
}

function kvkCheckRelaunch($kvkDepartments, $token)
{
    return executeSQL('
        SELECT DISTINCT(KVK)
        FROM [CC_DM_CAADQ].[UBO_LIJST]
        WHERE [VESTIGING] IN (' . $kvkDepartments . ')
        AND [KVK] IS NOT NULL
    ', $token);
}

function getKvkNoSilverRecord($grNumber, $token)
{
    return executeSQL('
    SELECT
        LEFT(kvk.[Kamer_van_Koophandel_nummer], 8) as kvkNumber,
        kd.[REL_NR_GOUD] as goldenRecord,
        kvk.[Vestigingsnummer_KVK] as kvkDepartment,
        kvk.[Bedrijfsnaam] + \'||\' + kvk.[handelsnaam1] + \'||\'
            + kvk.[handelsnaam2] + \'||\' + kvk.[handelsnaam3] as companyNames,
```

```
        kvk.[Straatnaam] as streetName,
        kvk.[Huisnummer] as houseNumber,
        kvk.[Huisnummer_toevoeging] as houseNumberAddition,
        kvk.[Postcode] as zipCode,
        kvk.[Woonplaats] as city,
        kvk.[Telefoonnummer] as phoneNumber,
        stuff(kvk.[Webadres], 1, 4, '') as website,
        kvk.[Branchecode_omschrijving] as activityDescription,
        kvk.[Datum_oprichting] as incorporationDate
    FROM [CC_DM_CAADQ].[KVK] as kvk
    LEFT OUTER JOIN [CC_DM_CAADQ].[KLANT_DETAIL] as kd
        ON LEFT(kvk.[Kamer_van_Koophandel_nummer], 8) = kd.[KVK_NR]
    WHERE kd.[KVK_NR] = (
        SELECT DISTINCT([KVK])
        FROM [CC_DM_CAADQ].[UBO_LIJST]
        WHERE [BEDRIJF_AUGRP] = 'Z036'
        AND [BEDRIJF_GOUD] = '0' . $grNumber . '\'
    )
    AND kd.[REL_NR_GOUD] = '' . $grNumber . '', $token);
}

function getUbo($kvkNumber, $searchForKvK, $token)
{
    if ($searchForKvK) {
        return executeSQL('SELECT
            [REL_NR_GOUD] as goldenRecord,
            [INITIALEN] as initials,
            [NAAM] as lastName,
            [STRAAT] as streetName,
            [HUIS_NR] as houseNumber,
            [HUIS_NR_TOEV] as houseNumberAddition,
            [POSTCODE] as zipCode,
            [PLAATS] as city,
            [EMAILADRES] as email,
            [TELNR_PRIVE] as phone,
            [TELNR_MOBIEL] as mobilePhone
        FROM [CC_DM_CAADQ].[KLANT_DETAIL]
        WHERE [REL_NR_GOUD] IN (
            SELECT DISTINCT RIGHT([UBO_GOUD], 9) as REL_NR_GOUD
            FROM [CC_DM_CAADQ].[UBO_LIJST]
            WHERE [KVK] = '' . $kvkNumber . '\'
            AND [BEDRIJF_AUGRP] = 'Z036'
        )
    ', $token);
    }

    return executeSQL('SELECT
        [REL_NR_GOUD] as goldenRecord,
        [INITIALEN] as initials,
        [NAAM] as lastName,
        [STRAAT] as streetName,
        [HUIS_NR] as houseNumber,
        [HUIS_NR_TOEV] as houseNumberAddition,
        [POSTCODE] as zipCode,
```

```
        [PLAATS] as city,
        [EMAILADRES] as email,
        [TELNR_PRIVE] as phone,
        [TELNR_MOBIEL] as mobilePhone
    FROM [CC_DM_CAADQ].[KLANT_DETAIL]
    WHERE [REL_NR_GOUD] IN (
        SELECT DISTINCT RIGHT([UBO_GOUD], 9) as REL_NR_GOUD
        FROM [CC_DM_CAADQ].[UBO_LIJST]
        WHERE [UBO_GOUD] = \'0' . $kvkNumber . '\'
        AND [BEDRIJF_AUGRP] = \'Z036\'
    )
  ', $token);
}
```

SAPBp.php

```
<?php
/*
All Sap BP related queries will be pulled from here.
*/

function sapGetChilds($grNumber, $token)
{
    return executeSQL('
        SELECT DISTINCT([silverRecord])
          ,[goldenRecord]
          ,[vta]
          ,[companyName]
          ,[kvkNumber]
          ,[kvkDepartment]
          ,[iban]
          ,[streetName]
          ,[houseNumber]
          ,[houseNumberAddition]
          ,[zipCode]
          ,[city]
          ,[postStreetName]
          ,[postHouseNumber]
          ,[postHouseNumberAddition]
          ,[postZipCode]
          ,[postCity]
          ,[postCountryIso]
          ,[phoneNumber]
          ,[email]
          ,[relationType] FROM [CC_DM_CAADQ].[DQ_SR_DETAILS]
        WHERE [goldenRecord] = \'' . $grNumber . '\'
    ', $token);
}

function sapMasterSearch($query, $srNumbers, $grNumber, $token)
{
    if ($srNumbers == '') {
        return executeSQL('
            SELECT DISTINCT([silverRecord])
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣,[goldenRecord]
␣␣␣␣␣␣␣␣␣␣␣␣␣␣,[vta]
␣␣␣␣␣␣␣␣␣␣␣␣␣␣,[companyName]
␣␣␣␣␣␣␣␣␣␣␣␣␣␣,[kvkNumber]
␣␣␣␣␣␣␣␣␣␣␣␣␣␣,[kvkDepartment]
␣␣␣␣␣␣␣␣␣␣␣␣␣␣,[iban]
␣␣␣␣␣␣␣␣␣␣␣␣␣␣,[streetName]
␣␣␣␣␣␣␣␣␣␣␣␣␣␣,[houseNumber]
␣␣␣␣␣␣␣␣␣␣␣␣␣␣,[houseNumberAddition]
␣␣␣␣␣␣␣␣␣␣␣␣␣␣,[zipCode]
␣␣␣␣␣␣␣␣␣␣␣␣␣␣,[city]
␣␣␣␣␣␣␣␣␣␣␣␣␣␣,[postStreetName]
␣␣␣␣␣␣␣␣␣␣␣␣␣␣,[postHouseNumber]
␣␣␣␣␣␣␣␣␣␣␣␣␣␣,[postHouseNumberAddition]
␣␣␣␣␣␣␣␣␣␣␣␣␣␣,[postZipCode]
␣␣␣␣␣␣␣␣␣␣␣␣␣␣,[postCity]
␣␣␣␣␣␣␣␣␣␣␣␣␣␣,[postCountryIso]
␣␣␣␣␣␣␣␣␣␣␣␣␣␣,[phoneNumber]
␣␣␣␣␣␣␣␣␣␣␣␣␣␣,[email]
␣␣␣␣␣␣␣␣␣␣␣␣␣␣,[relationType]␣FROM␣[CC_DM_CAADQ].[DQ_SR_DETAILS]
␣␣␣␣␣␣␣␣␣␣␣␣WHERE␣[companyName]␣!=␣\'\'
␣␣␣␣␣␣␣␣␣␣␣␣AND␣[relationType]␣=␣\'B\'
␣␣␣␣␣␣␣␣␣␣␣␣AND␣[goldenRecord]␣!=␣\'' . $grNumber . '\'
␣␣␣␣␣␣␣␣␣␣␣␣AND␣(' . $query . ')
␣␣␣␣␣␣␣␣␣␣␣␣ORDER␣BY␣[companyName],[goldenRecord];
␣␣␣␣␣␣␣␣', $token);
    }

    return executeSQL('
␣␣␣␣␣␣␣␣SELECT␣DISTINCT([silverRecord])
␣␣␣␣␣␣␣␣␣␣␣,[goldenRecord]
␣␣␣␣␣␣␣␣␣␣␣,[vta]
␣␣␣␣␣␣␣␣␣␣␣,[companyName]
␣␣␣␣␣␣␣␣␣␣␣,[kvkNumber]
␣␣␣␣␣␣␣␣␣␣␣,[kvkDepartment]
␣␣␣␣␣␣␣␣␣␣␣,[iban]
␣␣␣␣␣␣␣␣␣␣␣,[streetName]
␣␣␣␣␣␣␣␣␣␣␣,[houseNumber]
␣␣␣␣␣␣␣␣␣␣␣,[houseNumberAddition]
␣␣␣␣␣␣␣␣␣␣␣,[zipCode]
␣␣␣␣␣␣␣␣␣␣␣,[city]
␣␣␣␣␣␣␣␣␣␣␣,[postStreetName]
␣␣␣␣␣␣␣␣␣␣␣,[postHouseNumber]
␣␣␣␣␣␣␣␣␣␣␣,[postHouseNumberAddition]
␣␣␣␣␣␣␣␣␣␣␣,[postZipCode]
␣␣␣␣␣␣␣␣␣␣␣,[postCity]
␣␣␣␣␣␣␣␣␣␣␣,[postCountryIso]
␣␣␣␣␣␣␣␣␣␣␣,[phoneNumber]
␣␣␣␣␣␣␣␣␣␣␣,[email]
␣␣␣␣␣␣␣␣␣␣␣,[relationType]␣FROM␣[CC_DM_CAADQ].[DQ_SR_DETAILS]
␣␣␣␣␣␣␣␣WHERE␣[silverRecord]␣NOT␣IN␣(' . $srNumbers . ')
␣␣␣␣␣␣␣␣AND␣[companyName]␣!=␣\'\'
␣␣␣␣␣␣␣␣AND␣[relationType]␣=␣\'B\'
␣␣␣␣␣␣␣␣AND␣[goldenRecord]␣!=␣\'' . $grNumber . '\'
```

```
        AND (' . $query . ')
        ORDER BY [companyName],[goldenRecord];
    ', $token);
}

function checkFixed($grNumber, $token)
{
    return executeSQL('
        SELECT [REL_NR_GOUD]
        FROM [CC_DM_CAADQ].[KLANT_DETAIL]
        WHERE [REL_NR_GOUD] = \'' . $grNumber . '\'
        AND ([cleanse_FIXATIE_KANAAL] = \'data cleaning\'
        OR [cleanse_FIXATIE_KANAAL] = \'Data Cleaning\'
        OR [cleanse_FIXATIE_KANAAL] = \'Data cleaning\'
        OR [cleanse_FIXATIE_KANAAL] = \'Datacleaning\'
        OR [cleanse_FIXATIE_KANAAL] = \'datacleaning\')
    ', $token);
}
```

updatePlacedByEmp.php

```
<?php
/*
This code will be runed once a week to update the target value in
the machine learning table.
*/

include_once 'azure.php';
include_once 'sapBp.php';

ini_set('display_errors', 1);
ini_set('display_startup_errors', 1);
error_reporting(E_ALL);

//Azure authentication token
$token = $_GET['token'];

//get selection of not reviewed records, as all takes to long to run
$result = executeSQL('SELECT TOP(2000) FROM CC_DM_CAADQ.DQ_SR_ML
    WHERE [placedByEmp] IS NULL ORDER BY [grNumber] ASC', $token);

$grChild = '';
$skip = '';
$currentGr = 0;
$query = '';

foreach ($result as $ml) {
    if ($ml['grNumber'] != $currentGr) {
        $currentGr = $ml['grNumber'];
        $grChild = '';

        if (count(checkFixed($ml['grNumber'], $token)) === 0) {
            if (empty($ml['grNumberKvk']) || $ml['grNumberKvk'] == $ml['grNumber']
                || count(checkFixed($ml['grNumberKvk'], $token)) === 0) {
```

```
                $query .= 'UPDATE␣CC_DM_CAADQ.DQ_SR_ML␣SET␣placedByEmp␣=␣9
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣WHERE␣grNumber␣=␣\'' . $ml['grNumber'] . '\';';
                $skip = $ml['grNumber'];
                continue;
            }
        }
        $skip = '';

        //set all target values to 0
        $query .= 'UPDATE␣CC_DM_CAADQ.DQ_SR_ML␣SET␣placedByEmp␣=␣0
␣␣␣␣␣␣␣␣␣␣␣␣WHERE␣grNumber␣=␣\'' . $ml['grNumber'] . '\';';

        //get silverRecords placed by employee
        $grChild = sapGetChilds($ml['grNumber'], $token);
        if ($ml['grNumber'] !== $ml['grNumberKvk']
            && count($ml['grNumberKvk']) > 1) {
            $grChildKvk = sapGetChilds($ml['grNumberKvk'], $token);

            foreach ($grChildKvk as $record) {
                array_push($grChild, $record);
            }
        }

        if (count($grChild) === 0) {
            $skip = $ml['grNumber'];
            $query .= 'UPDATE␣CC_DM_CAADQ.DQ_SR_ML␣SET␣placedByEmp␣=␣9
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣WHERE␣grNumber␣=␣\'' . $ml['grNumber'] . '\';';
            continue;
        }
    }

    //if not fixed, continue to next gr object
    if ($ml['grNumber'] == $skip || $ml['grNumberKvk'] == $skip) {
        continue;
    }

    //checked if placed manually by employee
    if ($ml['grNumberKvk'] == $grChild[0]['goldenRecord']
        || $ml['grNumber'] == $grChild[0]['goldenRecord']) {
        foreach ($grChild as $child) {
            //if employee found same result, set target value to 1
            if (substr($child['silverRecord'], 1) == $ml['srNumber']) {
                $query .= 'UPDATE␣CC_DM_CAADQ.DQ_SR_ML
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣SET␣placedByEmp␣=␣1␣WHERE␣grNumber␣=␣\'' . $ml['grNumber'] .
                    '\'␣AND␣grNumberKvk␣=␣\'' . $ml['grNumberKvk'] .
                    '\'␣AND␣srNumber␣=␣\'' . $ml['srNumber'] . '\';';
            }
        }
    }
}

//last value probably not fully reviewed, so ignore and take with next run
$query .= 'UPDATE␣CC_DM_CAADQ.DQ_SR_ML␣SET␣placedByEmp␣=␣NULL
␣␣␣␣WHERE␣grNumber␣=␣\'' . $currentGr . '\';';
```

```
executeSQL($query, $token);
```