PST, AG
Frauentorstraße 3
Weimar, Germany
D - 99423 Weimar
.

July 15, 2020

OFAC Rules Implementation with ArangoDB

Luis Ramos

Document Version: 01-01

Contents

1 Changes . . . 3
2 Report Scope . . . 4
3 Dataset Characterization . . . 4
4 Ownership Algorithm . . . 7
5 OFAC Rules Implementation . . . 12
5.1 OFAC 100 % . . . 12
5.2 OFAC 50 % . . . 12
5.3 OFAC by Aggregation . . . 13
5.4 Ofac Ownership Relation Data Model . . . 13
6 Worst Case Scenarios . . . 15
7 Conclusions . . . 17

A List of Queries **19**

B Ownership Output **31**

C OFAC Santioned Ownerships **32**

D Worst Case Scenario Results **33**

1 Changes

Date	Author	Description
July 15, 2020	Luis Ramos	Initial version, and first exploration of dataset

Address
PST AG
Frauentorstraße 3
Weimar, Germany
99423

Board
Peter Piehler

Chairman of the Supervisory Board
Dr. Eberhard Piehler

Bank Details
Sparkasse Jena
Account 31062 BLZ 830 530 30

Tax Number
161/100/04145

Ust-IdNr DE 255 835 30

District Court
Weimar

HRB 501825

Telephone
+49(0)3641/8761070

Fax +49(0) 3641/8761071

Website
www.pst.ag

E-Mail
info@pst.ag

2 Report Scope

The Office of Foreign Assets Control (OFAC), since its inception, is a financial intelligence office with the responsibility of administrating economic and trade sanctions in support of the Government of the United States of America (USA), and its national security policies.

OFAC organizes its sanctions in lists, being the Specially Designated Nationals And Blocked Persons List (SDN) one of those. This list mentions individuals and companies that the USA considers are **owned by** or **controlled by**, or **acting for** or **on behalf of**, target countries. Moreover, it also include other non country specific organizations such as narcotic trafficker or terrorists.

The USA by means of OFAC also pretends to block all property, tangible or intangible, of sanctioned persons, where these sanctioned persons could have legitimate, direct or indirect interest. In this vein, OFAC published a guidance to illustrate the criteria to follow in order to define when a property, of a sanctioned person, must be sanctioned as well [1], thus the criteria to be used in this report is as follows:

- Properties belonging in 100 % of shares to a sanctioned person, must be sanctioned.
- Properties belonging to a sanctioned person in a quantity of shares more than or equal to 50 %, must be sanctioned.
- Properties belonging to more than one sanctioned persons, in a quantity of shares in more than or equal to 50 %, must be sanctioned.

Additional to the listed criteria description, in the frequently ask questions[2] of OFAC, **questions 401, 402** include a list of examples that I encourage to check.

As scope of this report we present the result of searching sanctioned companies, which address country is the Russian Federation, against the provided database, and identifying its punishable properties according to OFAC criteria, using the Business Register Number provided by Russia Federal Tax Registry (EGRUL), as keyword. Additionally, we profiled and defined some Worst Cases Scenarios, where OFAC rule were not fully applicable because of insufficient information.

3 Dataset Characterization

The dataset where we have to perform the search corresponds to **10,898,444** companies registered in EGRUL, and it is identified as `egrul_data_collection` in ArangoDB. Every company in this Dataset mostly, when apply,

[1] https://www.treasury.gov/resource-center/sanctions/Documents/licensing_guidance.pdf
[2] https://www.treasury.gov/resource-center/faqs/Sanctions/Pages/faq_general.aspx#50_percent

contains the following information:

- Names
- Addresses
- Registration
- Liquidation (- if applicable)
- Taxpayer registration details
- Company capital
- Official representative
- Shareholders
- Branch offices

Given that, as we mentioned in previous Section, the data is provided by EGRUL, we identified two possible keywords, those were the **Business Register Number** and **TaxID**. But, we needed to know whether or not every company could be identified by any of those numbers, thus we executed **query** 1 and **query** 2 on `egrul_data_collection`, obtaining the following results:

- 10,898,444 records with a **Business Register Number**
- 10,862,801 records with a **Tax ID Number**

This result indicates us that all documents in the datset can be identified by a **Business Register Number**, while there are 35,643 companies without **Tax ID Number**. However, we aim not only at finding specific companies, but to identify their owners, and their shares, consequently we also profiled the data in function of owners, executing queries **query** 3, **query** 4 and **query** 5, obtaining the following results:

- 9,614,346 records of companies with some **owner or founder**
- 1,284,089 records of companies with null **owner or founder**
- 9 records of companies with blank **owner or founder**

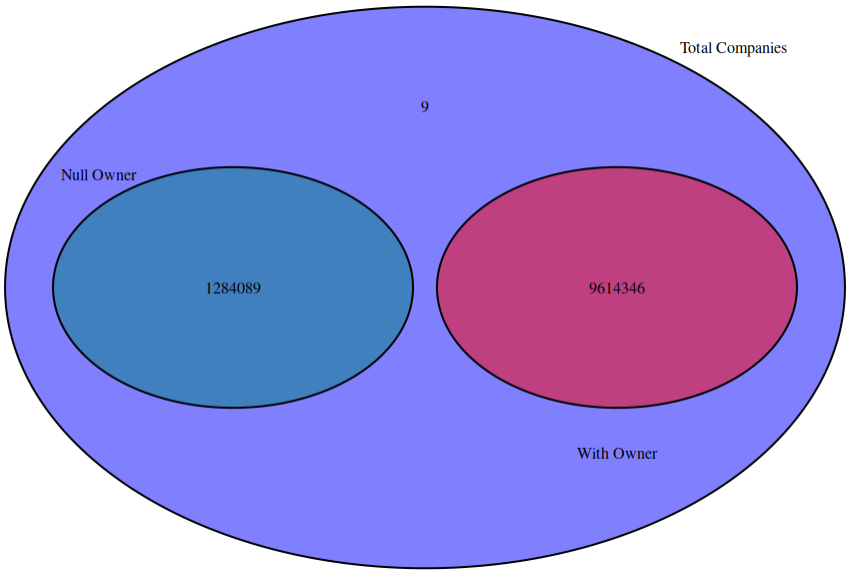

Figure 1: Information About Total Number of Founders

I consider important to highlight that **query** 5 was included, because the addition of resulting number of member in the subsets obtained from the execution of **query** 3 and **query** 4 was not equal to the whole. Figure 1 illustrate the scenario, that I explain in details as follows:

Consider we have a set of infinite many sets:

$A = \{b_0, b_1, \ldots, b_n\}$

Thus the union of those many sets, which constitute the main set, can be represented as follows:

$$\bigcup_{A} = \bigcup_{i} b_i \tag{1}$$

For our case, the addition of the **Null Owner** set and the **With Owner** set was not equal to the **Total Companies** set, making necessary to determine where was the difference, so could we could satisfy equation 1.

In the dataset wit owner we can identified the following owners:

- Founder Individual (FI)
- Investment Found (IF)
- Founder Municipality (FM)
- Founder (F)

- Founder Foreigner (FF)

I calculated that this quantity of owners could generate $2^5 = 32$ different sets, and to determine the number of elements (if any) per every set, individuals queries had to be built, and executed.

Normally, the members of those sets can be identified by the expression AB, that is the AND set operation, which in our case would implies several AND filters, one per every additional member of the set which, as I just said, could have up to 5 members in 32 combinations. However, we have the requirement of indexation, given the size of this data set. In this vein, according to ArangoDB user manual[3], when we use the AND operator indexing, the system does not consider more than one index, so it is recommended to work with the OR operator. Based on this shortcoming, I had to built queries based on the expression $\overline{\overline{A} + \overline{B}}$, and execute query 6, will the many combinations, on the dataset.

Figure 2 illustrates the resulting output and the characterization of owners subsets. From this, now our main task consist in apply OFAC rules in all subsets where members of **F** subset are present. But to apply such rules, first we need to know the ownership relation between companies, such procedure is described in next Section.

4 Ownership Algorithm

The dataset of sanctioned companies is identified as `ussdn_russian_id_collection` in ArangoDB. It consist of 213 records, and every record contains identifications numbers of companies, such as Company Register or tax ID. Given that ArangoDB is a GraphDB, the ownership relation between `ussdn_russian_id_collection` and `egrul_data_collection` was established by a collection of edges, named as `ofac_ownership_relation`. The process to get and store these ownership relations is described in Figure 3.

According to the algorithm, for every company in `egrul_data_collection`, we have to, at first, verify if every company has a *founder* as owner. If not the case, we proceed to next company, if it is the case, we store the Business Register Number of the *founder* Company and search it in `ofac_ownership_relation` as *owned*. If we do not find it, then we proceed to next company in `egrul_data_collection`, but if we find it, then we have to build a tuple with the Business Register Number of the *founder* (ownerid) and the Business Register Number of the owned company, thus we will have a tuple (ownerid, ownedi). This tuple is stored in `ofac_ownership_relation`.

Every time we finish a search on `egrul_data_collection` we have to count the occurrence of ownership, in other words, we have to count every time a new record is added to `ofac_ownership_relation`. When this value is equal to 0, the algorithm must be stopped. If in any iteration the value is non 0, we have to set $Occ = 0$, and repeat the procedure, until $Occ = 0$, which means no new owner was found.

[3] https://www.arangodb.com/docs/stable/indexing-index-utilization.html

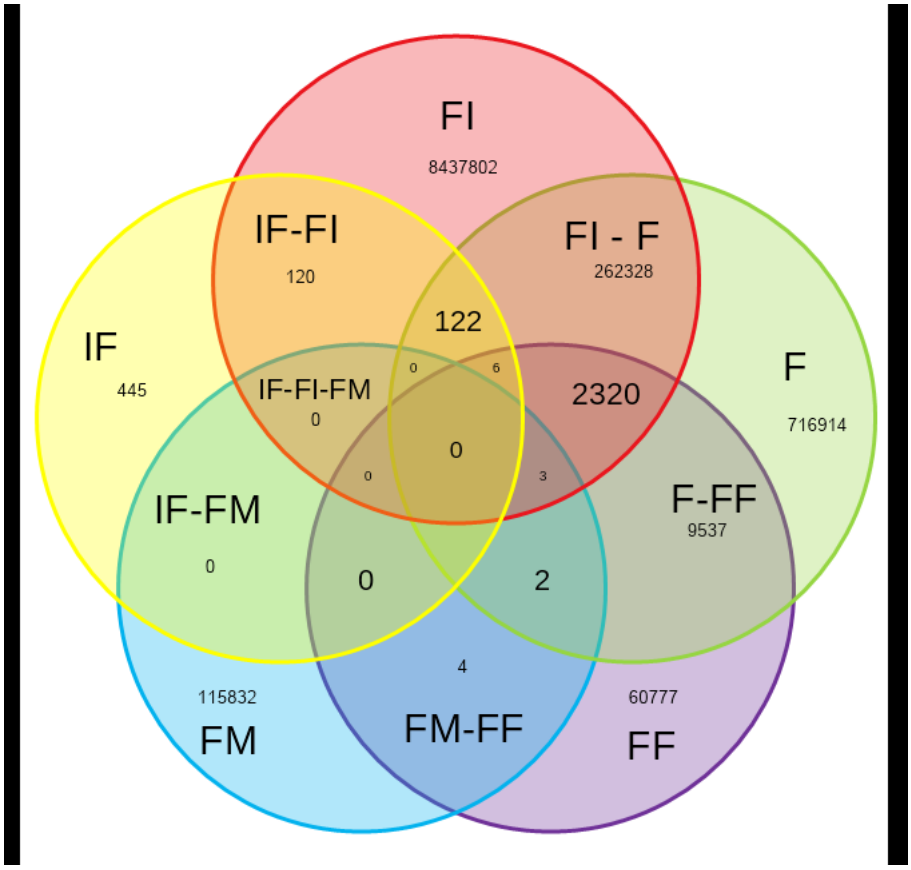

Figure 2: Information About Total Number of Founders

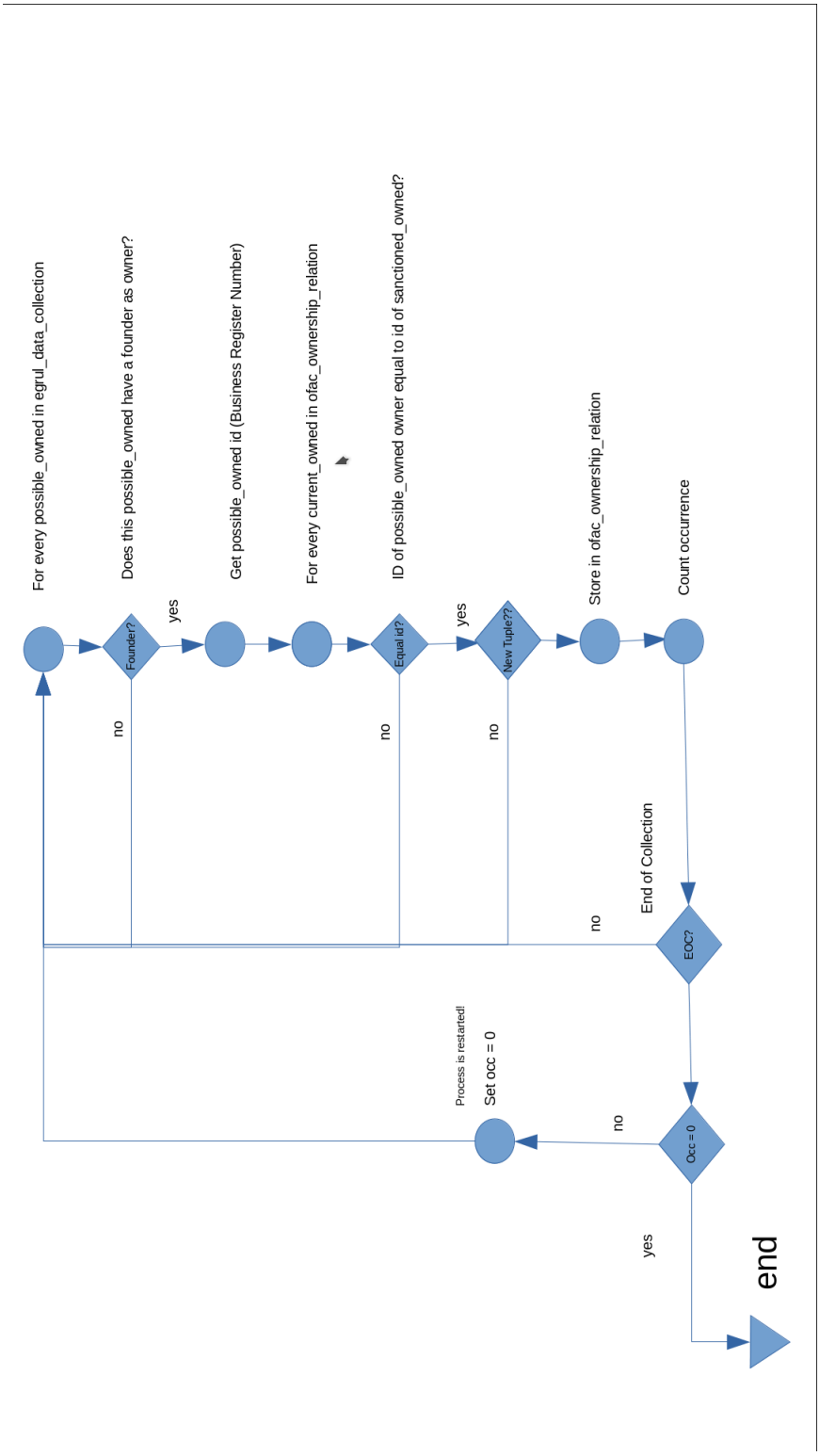

Figure 3: Algorithm for Ownership Relations

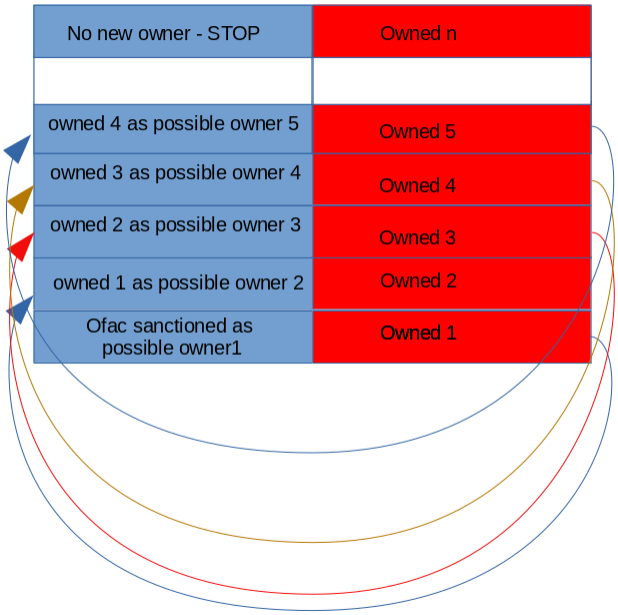

Figure 4: Ownership Stack

The result of implementing this algorithm can be visual represented as in Figure 4, where the stack grows as new ownership relations appear.

Query 7 is introduced as part of the algorithm depicted in Figure 3, and I consider worthy th remark that Query 7 must be applied recursively every time it finishes, if new owners were previously added to `ofac_ownership_relation` stack. Table B list all iterations performed, and the number of ownership relations found.

With respect to Query 7, I highlighted the LIMIT, because if we do not include a limit value, the query did not finish its execution in a reasonable time.

With regard to the execution time of this query, Figure 5 summarize this time. There it can be seen that for the first iteration, the execution time was around 1800 seconds, or 30 minutes. For the most iteration, execution time was between 30 and 45 minutes. Furthermore, we also inspected the frequency distribution of owners, that is we grouped companies by number of owners of different kinds. The result of this data inspection is presented in Figure 6. From it, we can evidently conclude that the most companies of this iteration have between one to two owners, while companies with more than 3 owners have low frequency. This analysis of frequency required additionally the use of the frequency function of a datasheet processor (Excel).

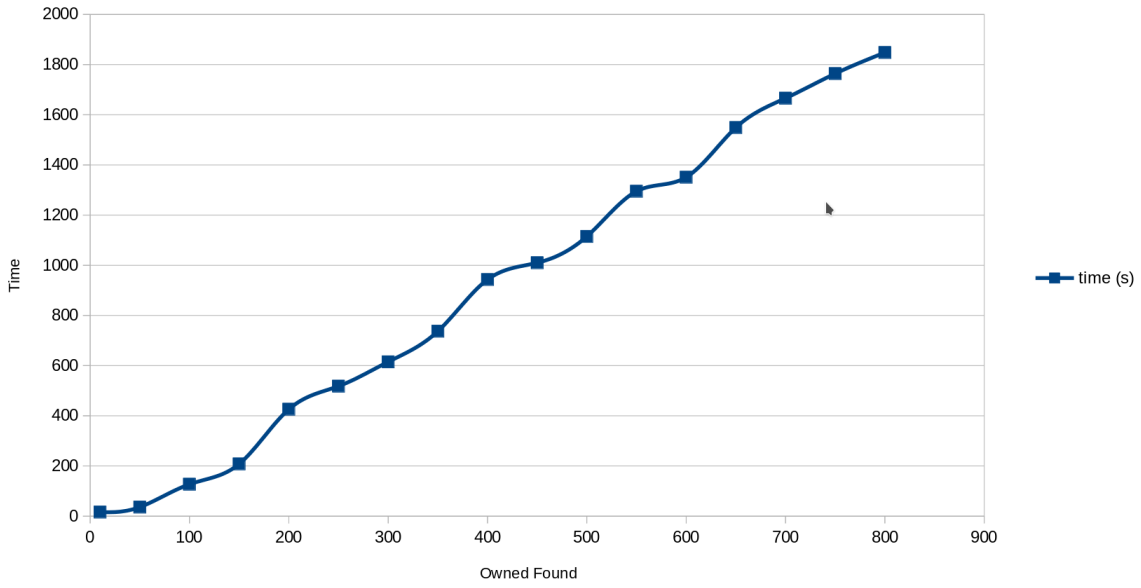

Figure 5: Performance of Ownership Query

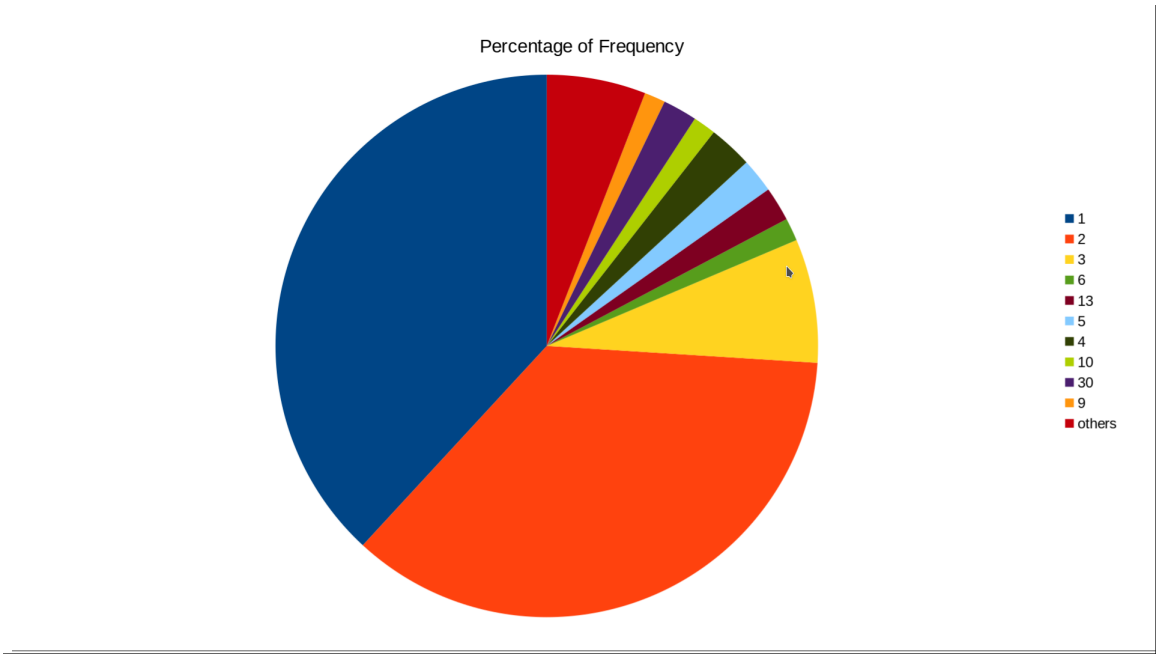

Figure 6: Frequency Distribution of Ownership

5 OFAC Rules Implementation

In Section 2, we declared that the our main objective was to apply the OFAC rules against the dataset represented by `egrul_data_collection` (See Section 3). For such a task it was necessary to previously create a network of ownership relations between companies (`ofac_ownership_relation`). Such procedure was described in Section 4, where a collection of relations was generated. In this Section we present the result of applying the OFAC rules in the provided dataset using the generated ownership relations. All reference to OFAC sanctions, what we will do next, were recorded in `ofac_ownership_relation` collection.

5.1 OFAC 100 %

The OFAC 100 % rule can be explained as we just indicated in Section 2, with the following proposition:

- Properties belonging in 100 % of shares to a sanctioned person, must be sanctioned.

This rule can be represented in ArangoDB Query Language (AQL) as shown in Query 8. This query was executed on `egrul_data_collection` as we just explained, and the results were recorded in Table C, located in Appendix D. To the date, there is a total of **238** companies that belong in 100 % to a company directly sanctioned by OFAC, (Iteration 1), or to a company owned by a sanctioned company (further iterations), and which fulfill the OFAC sanction rules criteria. It may be noted that while in Table B there are 25 iterations, in Table C there were 5 iteration with the query corresponding to OFAC 100 %. That difference in iterations occurs not only because they correspond to different cases, but because in case of OFAC 100 % the procedure was stopped when no new sanctioned company was found in the iteration.

5.2 OFAC 50 %

The OFAC 50 % rule can be explained as we just indicated in Section 2, with the following proposition:

- Properties belonging to a sanctioned person in a quantity of shares more than or equal to 50 %, must be sanctioned.

The rule mentioned above can be represented in AQL as illustrated in Query 9. We followed the same procedure indicated in previous Subsection, and included the results in the same Table (Table C, located in Appendix D). With the application of this query we flagged **93** ownership relations as sanctioned by OFAC 50 %.

5.3 OFAC by Aggregation

The OFAC by Aggregation (AGG) rule can be explained as we just indicated in Section 2, with the following proposition:

- Properties belonging to more than one sanctioned person, in a quantity of aggregated shares in more than or equal to 50 %, must be sanctioned.

This rule differs from the two previous ones in the sense that the evaluation of a owned company have the precondition of having more than one owner, and that no owner must hold more than 50 % of shares, the latter means that the reach of 50 % must be the result of adding the share of at least 2 companies. Figure 7 presents an illustrative relation of an arbitrary owned company Oed_n and a group of owners represented as $\{Oer_0, Oer_{ss_0}, Oer_{ss_1}, Oer_{ss_2}, \ldots, Oer_{ss_n}\}$. It is evident that there are two kind of owners, those with **Sanctioned Shares** (ss_n), and some others without **Sanctioned Shares**. In this point we want to remark that every arrow in the Figure represents an ownership relation, and that there is a direction of the ownership (from, to), which facilitates the execution of the algorithm, and the evaluation of every member of the network. It is also worthy to mention that the ownership evaluation occurs strictly in the first ring around the owned company.

The just explained notions can be formalized as follows for every Oed_n or, what is the same, for every owned node, in `egrul_data_collection`:

$$ofac_sanctioned = \begin{cases} \sum_{i=0}^{n} ss_i, & if \sum_{i=0}^{n} ss_i \geq 50 \text{ and } ss_i < 50 \text{ and n} > 1 \\ null, & else \end{cases} \tag{2}$$

Equation 2 summarizes the case of flagging a company as sanctioned by aggregation. In this case ss_i corresponds to the **Sanctioned Shares** of every company, if any. That means only the shares of sanctioned owners will be considered and added, moreover the sum of sanctioned shares must be greater or equal to 50, the number of sanctioned shares of any company must not be greater than 50 ($ss_i < 50$), and the number of sanctioned owners must be more that 2 (in this case we wrote $n > 1$ because n count begins in 0). In the case that any of those conditions is not fulfilled, this equation must resolve to **null**. In other words, and for practical applications, no writing action must occur in `ofac_ownership_relation`.

5.4 Ofac Ownership Relation Data Model

In Listing 1 we present the current data model we are currently following to build up `ofac_ownership_relation` collection. It consists of two internal identifiers for ArangoDB (lines 2 and 3), two identifiers provided by `egrul_data_collection`,

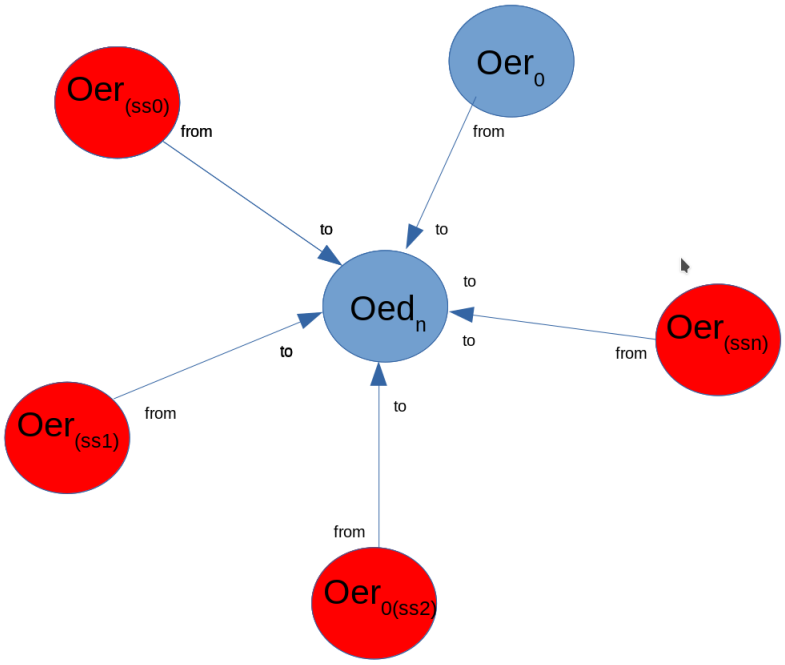

Figure 7: Sanctioned by Aggregation

the `ofac_sanctioned` attribute, that provides the kind of sanction, if any. That is 100 for ofac 100 %, 50 for ofac 50 %, AGG for ofac by aggregation, or WSC for Worst Case Scenario, which will be explained in next Section. The last component of this model is `ofac_status` which contains the legal status of the company, which can be active, inactive, or terminated.

Listing 1: Ofac Ownership Relation Data Model

```
{
        "_from": "ArangoDB ID of an ofac santioned company",
        "_to": "ArangoDB ID of a company owned by an ofac santioned company",
        "owner_businessRegisterNumber": "a businessRegisterNumber",
        "owned_businessRegisterNumber": "a businessRegisterNumber",
        "ofac_santioned": "if any, only a value from 100, 50, AGG, WCS",
        "ofac_status": "if any, terminated, active or inactive"
}
```

6 Worst Case Scenarios

If we compare the number of companies with no ownership attribute (null owners) reported in Figure 1, which is 1,284,089; against the number of founders, and the combination of ownerships that include at least one founder, which number is 991,099, as illustrated in Figure 2, we could lucubrate that because of the dependency of the OFAC sanctions on the shares information, in some cases this information could be hidden or not provided.

In this order of ideas, we elaborated some Worst Case Scenario (WCS) related with this lack of information about shares, which are not exhaustive, or limiting, but could give some guides about how to proceed:

1. A company owned by a direct sanctioned company, this sanctioned company is the only one owner presented, but there with no share information in the file.

2. A company owned by a direct sanctioned company, many different owners, but no share information from any owner.

3. A company owned by a direct sanctioned company, many different owners, but shares information is not complete. The failing shares plus the sanctioned shares can generate a sanctioned company.

The first scenario is the most committed, because if there were shares information that confirms the sanctioned owner owns 100 % of the shares, we must proceed to add the sanction to this ownership relation. Query 11 represents

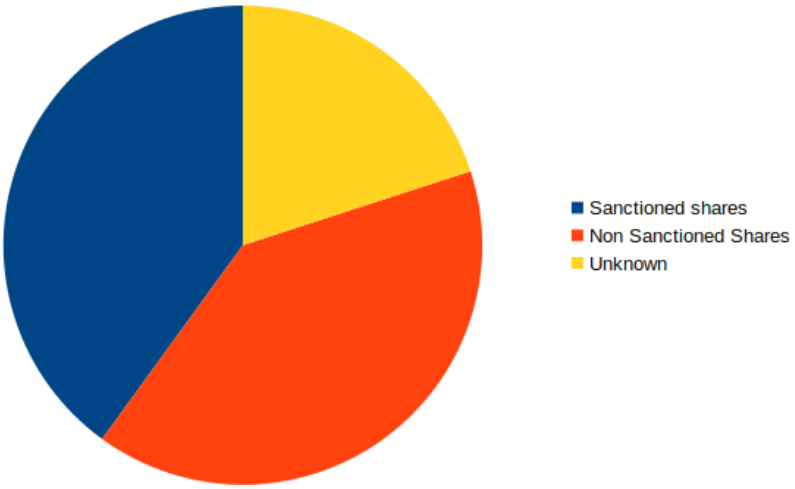

Figure 8: Worst Case Scenario

this requirement in AQL. The second WCS corresponds to the possibility of a company having many owners, all of them with no shares information, and one of those sanctioned. In that scenario the level of uncertain is very high because we do not have any guide to determine if any sanctioned pattern is applicable. This second WCS is found by running Query 12.

The last WCS can be illustrated with Figure 8. There, we can see that there are two large regions, and an smaller one. For this case we assume the larger ones are defined, that means one part can represent to a known sanction shares part, and the another part can represent to the known non sanctioned part of the shares, but no one of them has more than 50 % of the shares. While, there is a part which are undetermined or not assigned to any owner, thus we fall in an uncertain, because if it happens that these shares corresponds to a sanctioned company, we would be before the presence of a punishable company. Query 13 let us obtain such information, if any case, from `egrul_data_collection`.

In Table D we summarize the resul of executing queries 11, 12 and 13 on `egrul_data_collection`. In the case of WCS 1 and WCS 2, the procedure finished after two iterations. In case of WCS, the process finishes after the first iteration. In the column of results, there are two values for every iteration, when it applies. The first value correspond with the number of companies that fulfill the search pattern, and the second number corresponds to the part of those companies who to date are still actives.

One of the results that we could highlight is the fact that, in the WCS 1, from the 115 found, where only a sanctioned company appears as owner, but we do not have any shares information, 73 of them were terminated (63 %), what means they are not active any more, and no successor was declared. We could speculate that these 73 companies were closed because they actually belong in its whole to the sanctioned owner mentioned in the register

document. In the second iteration of this same WCS the percentage of closed companies is still higher (66 %), while in the other case, the percentage of closed companies is not so marked, actually in some cases no closed company was found in the iteration.

To finish this exposition we want to introduce Figure 9, where we show two graph. Every one corresponds to an iteration or query execution in WCS 1. On the graph of Iteration 1 we have circled 3 common patterns of that particular graph, where the ownership of one company propagates from 1 to many other companies. On the other hand, on Iteration 2 occurs exactly the inverse behavior, that is many companies participate in the ownership of only one company. Obviously, you may wonder whether or not this behavior has something to do with the fact that the former company, of which there is no shares information, belong to a sanctioned company. And that, at the same time, the later company belongs to these former ones.

To answer all those questions, firstly we should determine if such kind of information is relevant for the customer or final user, and further we require to determine which of these patterns can be considered as normal or regular in `egrul_data_collection`. For the case of finding patters, we will mentioned some usefull methodologies in next Section.

7 Conclusions

Results evident that AQL is enough to represent all OFAC rules, however current ArangoDB implementation has a time execution per query that is considerable large yet (30 minutes). Because of this algorithm described in Figure 3 was not finished, and we stopped it in iteration 25.

It is also worthy to remark that `egrul_data_collection` has 10 % of companies with not owners information, which makes not possible to evaluate these companies under OFAC rules. WCS were consider to try to offset the loss of information about shares in many companies.

In the absence of these information about shares, or in the case of wanting to delve into these WCS we recommend to evaluate the possibility to implement the Machine Learning (ML) approach. ML works in two possible configurations, supervised and non supervised learning. The different among those configuration is while the first requires an input pattern, the later does not. For instance, we could confirm the existence of the patterns shown in Figure 9 and its quantity on `egrul_data_collection` using the supervised configuration. While in the unsupervised configuration patterns are determined by the system itself. If we now what ownership patters are regular in `egrul_data_collection`, we can make better judgments about the patter we find in companies where OFAC sanctioned companies have shareholding.

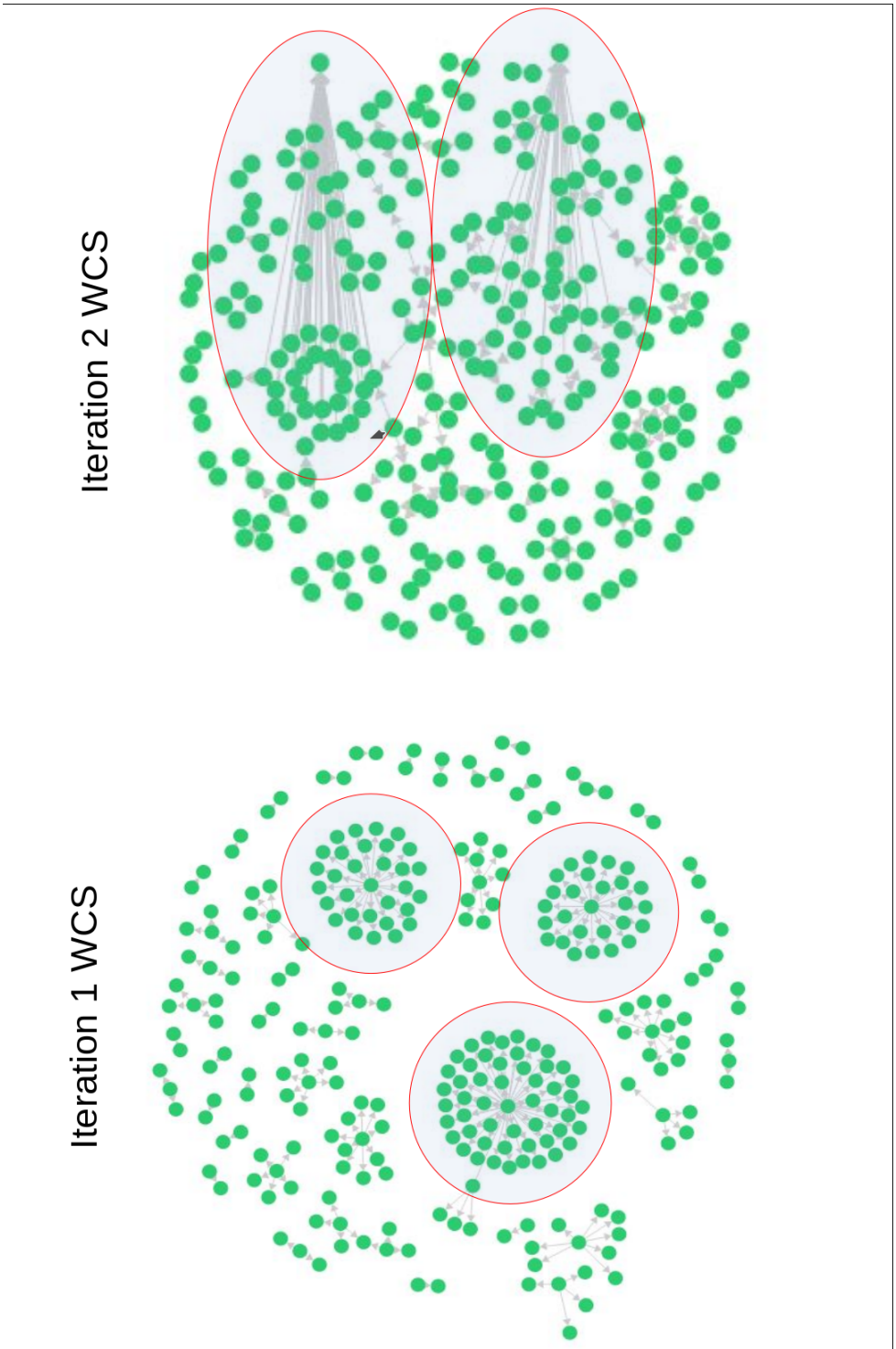

Figure 9: Graph WCS

Address
PST AG
Frauentorstraße 3
Weimar, Germany
99423

Board
Peter Piehler

Chairman of the Supervisory Board
Dr. Eberhard Piehler

Bank Details
Sparkasse Jena
Account 31062 BLZ 830 530 30

Tax Number
161/100/04145

Ust-IdNr DE 255 835 30

District Court
Weimar

HRB 501825

Telephone
+49(0)3641/8761070

Fax +49(0) 3641/8761071

Website
www.pst.ag

E-Mail
info@pst.ag

Appendix A

List of Queries

1. **get_businessRegisterNumber**

```
FOR doc IN egrul_data_collection
  FILTER doc.businessRegisterNumber.literal !=""
  COLLECT WITH COUNT INTO length
RETURN length
```

2. **get_taxID**

```
FOR doc IN egrul_data_collection
   FILTER doc.taxId.literal !=""
   COLLECT WITH COUNT INTO length
RETURN length
```

3. **get_informationAboutFounders_null**

```
FOR owner in egrul_data_collection
   FILTER owner.informationAboutFounders == null
RETURN   owner._id//owner.businessRegisterNumber//
```

4. **get_informationAboutFounders**

```
FOR owner in egrul_data_collection
   FILTER owner.informationAboutFounders != null
RETURN   owner._id//owner.businessRegisterNumber//
```

5. **get_informationAboutFounders_blank**

```
FOR owner in egrul_data_collection
   FILTER owner.informationAboutFounders == ""
RETURN   owner._id//owner.businessRegisterNumber//
```

6. **get_Owners_sets**

```
FOR owner in egrul_data_collection
    LET InvestmentFund = owner.informationAboutFounders.unitInvestmentFund
    LET founderIndividual = owner.informationAboutFounders.founderIndividual//
    LET founder = owner.informationAboutFounders.founder//
    LET founderForeign = owner.informationAboutFounders.founderForeign
    LET founderMunicipalty = owner.informationAboutFounders.founderMunicipalty
```

FILTER (!(founder != null or founderForeign != null or founderIndividual != null) or (!(founderMunicipalty != null or InvestmentFund !=null)))

```
    LIMIT 100
    COLLECT WITH COUNT INTO length
    LIMIT 10
RETURN length
```

The FILTER can be changed with some of the following combinations:

(a) ((InvestmentFund != null) or (founderIndividual != null)) or ((founder == null) or (founderForeign == null) or (founderMunicipalty == null))

(b) ((founder != null) or (InvestmentFund != null or (founderMunicipalty != null))) or ((founderForeign == null) or(founderIndividual == null))

(c) ((founderMunicipalty != null) or (founderForeign != null) or (InvestmentFund != null)) or ((founder == null) or (founderIndividual == null))

(d) ((founderForeign != null) or (InvestmentFund != null)) or ((founderMunicipalty == null) or (founder == null) or (founderIndividual == null))

(e) ((founderMunicipalty != null) or (InvestmentFund != null)) or ((founderForeign == null) or (founder == null) or (founderIndividual == null))

(f) ((founderMunicipalty == null) or (founderIndividual != null) or (founder != null)) or ((InvestmentFund == null) or (founderForeign == null))

(g) ((founderIndividual != null) or (founder != null)) or ((founderMunicipalty == null) or (InvestmentFund == null) or (founderForeign == null))

(h) ((founderIndividual != null) or (founder != null) or (founderForeign != null)) or ((founderMunicipalty == null) or (InvestmentFund == null))

(i) ((founderMunicipalty != null) or(founderIndividual != null) or (founder != null)) or ((InvestmentFund == null) or (founderForeign == null))

(j) ((founder != null) or (founderForeign != null)) or ((founderMunicipalty == null) or (founderIndividual == null) or (InvestmentFund == null))

(k) ((founderMunicipalty != null) or (founder != null)) or ((founderForeign == null) or (founderIndividual == null) or (InvestmentFund == null))

(l) ((founderMunicipalty != null) or (founderForeign != null)) or ((founder == null) or (founderIndividual == null) or (InvestmentFund == null))

(m) ((founderMunicipalty != null)) or ((founderForeign == null) or (founder == null) or (founderIndividual == null) or (InvestmentFund == null))

Address
PST AG
Frauentorstraße 3
Weimar, Germany
99423

Board
Peter Piehler
Chairman of the Supervisory Board
Dr. Eberhard Piehler

Bank Details
Sparkasse Jena
Account 31062 BLZ 830 530 30

Tax Number
161/100/04145
Ust-IdNr DE 255 835 30

District Court
Weimar
HRB 501825

Telephone
+49(0)3641/8761070
Fax +49(0) 3641/8761071

Website
www.pst.ag
E-Mail
info@pst.ag

7. **get_Ownership_Relation**

```
FOR owned in egrul_data_collection
    filter owned.informationAboutFounders.founder!=null
    LET founders = owned.informationAboutFounders.founder
    FILTER LENGTH(founders)  > 0
    FOR founder IN founders
        LET OwnerbusinessRegisterNumber = founder.nameAndId.businessRegisterNumber.literal
        FILTER OwnerbusinessRegisterNumber != null
        //now we compare with current ownership
        FOR sanctioned IN ofac_ownership_relation
            FILTER sanctioned.owned_businessRegisterNumber == OwnerbusinessRegisterNumber
            LET OwnedbusinessRegisterNumber = owned.businessRegisterNumber.literal
            LET ownedid = owned._id
            LET ownerid = CONCAT("egrul_data_collection/",OwnerbusinessRegisterNumber)
            //writing into collection  with error catch
            INSERT {_from:ownerid, _to:ownedid,
            "owned_businessRegisterNumber":OwnedbusinessRegisterNumber}
            INTO ofac_ownership_relation // OPTIONS { ignoreErrors: true }
```

LIMIT 1100

```
            COLLECT WITH COUNT INTO length
RETURN  {ownerid, OwnedbusinessRegisterNumber, ownedid}
```

8. **set_ofac_100**

```
//set ofac 100 %
//here we revise every santioned entity belong to only one owner,
//that this owner owned 100 % of actions,
//and that this owner is sanctioned, either by ofac 100, 50 or agg.
//if that is the case, we proceed to flag them as ofac_sanctioned 100%
//from the ring 2 we have to confirmed that ofac santioned attribute is not null,
// I mean we have to confirme previousl company is sanctioned
//getting id of entity in first collection

FOR doc1 IN ofac_ownership_relation
   LET doc1id = doc1._to
   LET ofac_ownerID = doc1._from
   LET ofac_ownerID2 = LTRIM(ofac_ownerID, "egrul_data_collection/")
   LET sanction_value = doc1.ofac_santioned
   FOR doc2 in egrul_data_collection
      LET doc2id = doc2._id
      FILTER doc1id == doc2id
      //with the company identify, we proceed to coount ownerships if any
      LET founder = doc2.informationAboutFounders.founder
          LET founder_size = length(founder)
```

```
        LET founderIndividual = doc2.informationAboutFounders.founderIndividual
        LET founderIndividual_size = length(founderIndividual)
        LET InvestmentFund = doc2.informationAboutFounders.unitInvestmentFund
        LET InvestmentFund_size = length(InvestmentFund)
        LET founderForeign  = doc2.informationAboutFounders.founderForeign
          LET founderForeign_size = length(founderForeign)
        LET founderMunicipalty = doc2.informationAboutFounders.founderMunicipalty
        LET founderMunicipalty_size = length(founderMunicipalty)
        //counting total amount of owners
        LET owners_count = founderIndividual_size+InvestmentFund_size+
        founderForeign_size+founderMunicipalty_size
        FILTER (owners_count == 0 and founder_size == 1)
        FOR doc3 IN founder
           LET share_size  = doc3.founderShare.sizeOfShares.sizeOfSharePercentage
           LET ownerID_json = doc3.nameAndId.businessRegisterNumber.literal
           //here we confirm two conditions:
           //Owner owns 100 % of the actions
           //Owner ID is the same in ofac ownership edges
           //collection and in egrul main collection.
           FILTER (share_size == 100 and ownerID_json == ofac_ownerID2)
           LET owner_ofac_array =
           (
           FOR doc4 IN ofac_ownership_relation
              LET doc4id = doc4.owned_businessRegisterNumber
              //LET doc4id2 = LTRIM(doc4id, "egrul_data_collection/")
              FILTER doc4id == ofac_ownerID2//sanctioned owner identified
              LET ofac_status = doc4.ofac_santioned
              //FILTER ofac_commited_share != null
              //COLLECT WITH COUNT INTO length
              RETURN ofac_status//ofac_commited_share//length
            )
        LET owner_ofac_value = owner_ofac_array[0]
        FILTER owner_ofac_value != null
        FILTER sanction_value == null
        UPDATE doc1 WITH {ofac_santioned: "100 %"} IN ofac_ownership_relation//
        LIMIT 1000
        COLLECT WITH COUNT INTO length
  RETURN    {doc1id,owner_ofac_value}//,length//
```

9. **set_ofac_50**

```
  //set ofac 50 %
  //here we revise every santioned entity,
  //visit the owners and confirme ownership value is more than or equal to 50 %
  //if that is the case, we proceed to flag them as ofac_sanctioned 100%
  //getting id of entity in first collection

  FOR doc1 IN ofac_ownership_relation//variable to change
     //this is the variable to change across edges
```

```
//this variable chage across set of edges
LET doc1id = doc1._to//identifier of owned
LET ofac_ownerID = doc1._from
LET ofac_ownerID2 = LTRIM(ofac_ownerID, "egrul_data_collection/")
LET sanction_value = doc1.ofac_santioned
FOR doc2 in egrul_data_collection
   LET doc2id = doc2._id
   FILTER doc1id == doc2id
   //with the company identify, we proceed to coount ownership
   LET founder = doc2.informationAboutFounders.founder
   //getting list of founder owners if any
   FILTER founder != null
   //we proceed to identify companies with more than 50 and less than 100 %
   //Note: it is possible than only one company owns less than 100 %,
   //althought no more owners appear
   FOR doc3 IN founder
      LET share_size  = doc3.founderShare.sizeOfShares.sizeOfSharePercentage
      LET ownerID_json = doc3.nameAndId.businessRegisterNumber.literal
      //here we confirm two conditions:
      //Owner owns 100 % of the actions
      //Owner ID is the same in ofac ownership
      //edges collection and in egrul main collection.
      FILTER ((share_size >= 50 and share_size < 100)
      and ownerID_json == ofac_ownerID2)//
      //checking owner in previous collection is sanctioned
      LET owner_ofac_array =
      (
         FOR doc4 IN ring4_ofac_ownership_relation//previous owner
            LET doc4id = doc4.owned_businessRegisterNumber
            FILTER doc4id == ofac_ownerID2//sanctioned owner identified
            LET ofac_status = doc4.ofac_santioned
            //COLLECT WITH COUNT INTO length
            RETURN ofac_status//ofac_commited_share//length
         )
      LET owner_ofac_value = owner_ofac_array[0]
      FILTER owner_ofac_value != null
      //filtering owners with exactly one sanctioned flag
      FILTER sanction_value == null
      //after confirming a company is sanctioned,
      //and owns at least 50 % of a company, we flag it
      UPDATE doc1 WITH {ofac_santioned: "50 %"} IN ofac_ownership_relation
      //variable to change
      LIMIT 1000
      COLLECT WITH COUNT INTO length
RETURN   {share_size, ownerID_json, ofac_ownerID2,   owner_ofac_value}
```

10. **set_ofac_aggregation**

```
//set ofac by aggregation from ring 2

FOR doc1 IN ofac_ownership_relation//this is the variable to change across edges
//this variable chage across set of edges
   LET doc1id = doc1._to//identifier of owned
   LET ofac_ownerID = doc1._from
   LET ofac_ownerID2 = LTRIM(ofac_ownerID, "egrul_data_collection/")
   LET sanction_value = doc1.ofac_santioned//
   FOR doc2 in egrul_data_collection
      LET doc2id = doc2._id
      FILTER doc1id == doc2id
      //we confirm we are searching the same owned document
      //with the company identify, we proceed to coount ownership
      LET founder = doc2.informationAboutFounders.founder
      LET founder_size = length(founder)
      //counting total amount of owners
      //LET owners_count = founder_size+
      founderIndividual_size+
      InvestmentFund_size+
      founderForeign_size+
      founderMunicipalty_size
      FILTER (founder_size > 1)//getting at least one owner
      //getting size of share in percentage
      LET ofac_commited_addition = (
      FOR doc3 IN founder
         LET share_size  = doc3.founderShare.sizeOfShares.sizeOfSharePercentage
         FILTER share_size != null
         LET ownerID_json = doc3.nameAndId.businessRegisterNumber.literal
         //given that we just know the size of the share for every company,
         //we confirm they are sactioned
         //with a subquery
            LET owner_ofac_array =
            (
            FOR doc4 IN  ofac_ownership_relation//previous relation link
               LET doc4id = doc4.owned_businessRegisterNumber
               FILTER doc4id == ownerID_json//ofac_ownerID2//sanctioned owner identified
               LET ofac_status = doc4.ofac_santioned
               FILTER ofac_status != null//here we confirm owner is sanctioned
            RETURN ofac_status//ofac_commited_share//length
            )
         LET owner_ofac = owner_ofac_array[0]
         FILTER owner_ofac != null
         //we check there is a sanction flag in owner
         COLLECT committed_share = share_size
         RETURN committed_share
         )//end second subquery
         LET sum_ofac_shares = SUM(ofac_commited_addition)
         FILTER sum_ofac_shares >= 50
```

Address
PST AG
Frauentorstraße 3
Weimar, Germany
99423

Board
Peter Piehler

Chairman of the Supervisory Board
Dr. Eberhard Piehler

Bank Details
Sparkasse Jena
Account 31062 BLZ 830 530 30

Tax Number
161/100/04145

Ust-IdNr DE 255 835 30

District Court
Weimar

HRB 501825

Telephone
+49(0)3641/8761070

Fax +49(0) 3641/8761071

Website
www.pst.ag

E-Mail
info@pst.ag

```
            FILTER sanction_value == null//granting that is not previoulsy sanctioned
            //with this information we can proceed to flag
            UPDATE doc1 WITH {ofac_santioned: "AGG"} IN ofac_ownership_relation//
RETURN      sum_ofac_shares//
```

11. **set_ofac_100_wcs**

```
FOR doc1 IN ofac_ownership_relation
   LET doc1id = doc1._to//identifier of owned
   LET ofac_ownerID = doc1._from//identifier of owner
   LET ofac_ownerID2 = LTRIM(ofac_ownerID, "egrul_data_collection/")
   //removing not needed string
   LET sanction_value = doc1.ofac_santioned//checking if there is previous sanction flag
      FOR doc2 in egrul_data_collection
         LET doc2id = doc2._id
         FILTER doc1id == doc2id
         LET founder = doc2.informationAboutFounders.founder
           LET founder_size = length(founder)
         LET founderIndividual = doc2.informationAboutFounders.founderIndividual
         LET founderIndividual_size = length(founderIndividual)
         LET InvestmentFund = doc2.informationAboutFounders.unitInvestmentFund
         LET InvestmentFund_size = length(InvestmentFund)
         LET founderForeign  = doc2.informationAboutFounders.founderForeign
         LET founderForeign_size = length(founderForeign)
         LET founderMunicipalty = doc2.informationAboutFounders.founderMunicipalty
         LET founderMunicipalty_size = length(founderMunicipalty)
         //counting total amount of owners
         LET owners_count = founderIndividual_size+
         InvestmentFund_size+
         founderForeign_size+
         founderMunicipalty_size
         FILTER (owners_count == 0 and founder_size == 1)//getting only one owner,
         and that owner has to be founder
         //getting size of share in percentage
         FOR doc3 IN founder
            LET share_size  = doc3.founderShare.sizeOfShares.sizeOfSharePercentage
            LET ownerID_json = doc3.nameAndId.businessRegisterNumber.literal
            //here we confirm two conditions:
            //Owner owns 100 % of the actions
            //Owner ID is the same in ofac ownership edges collection
            //and in egrul main collection.
            FILTER (share_size == 100 and ownerID_json == ofac_ownerID2)
            //we confirm owner id of JSON is in the ownership list
            //we make a sub query to check if owner has a sanction_value not null,
            then we proceed to sanction the child as well
            //NOTE: in first ring cllection this condition is not fulfilled,
            //so we have to ber carefull in his application
            LET owner_ofac_array =
```

```
            (
               FOR doc4 IN ring4_ofac_ownership_relation
               LET doc4id = doc4.owned_businessRegisterNumber
               //LET doc4id2 = LTRIM(doc4id, "egrul_data_collection/")
               FILTER doc4id == ofac_ownerID2//sanctioned owner identified
               LET ofac_status = doc4.ofac_santioned
               RETURN ofac_status//
               )
            LET owner_ofac_value = owner_ofac_array[0]
            FILTER owner_ofac_value != null
            FILTER sanction_value == null//here we grant to flag no flagged owned companies
            UPDATE doc1 WITH {ofac_santioned: "100 %"} IN ofac_ownership_relation//
            LIMIT 1000
      COLLECT WITH COUNT INTO length
RETURN    {doc1id,owner_ofac_value}
```

12. **set_wcs_02**

```
FOR doc1 IN ofac_ownership_relation
   LET doc1id = doc1._to//identifier of owned
   LET ofac_ownerID1 = doc1.owner_businessRegisterNumber//identifier of owner
   LET ofac_ownedID1 = doc1.owned_businessRegisterNumber//identifier of owner
   LET sanction_value = doc1.ofac_santioned//checking if there is previous sanction flag
      FOR doc2 in egrul_data_collection
      LET doc2id = doc2._id
      FILTER doc1id == doc2id
      LET businessRegisterNumber = doc2.businessRegisterNumber.literal
      //with the company identified, we proceed to coount ownerships if any
      LET founders = doc2.informationAboutFounders.founder
      LET founder_size = length(founders)
      LET founderIndividuals = doc2.informationAboutFounders.founderIndividual
      LET founderIndividual_size = length(founderIndividuals)
      LET InvestmentFunds = doc2.informationAboutFounders.unitInvestmentFund
      LET InvestmentFund_size = length(InvestmentFunds)
      LET foundersForeign  = doc2.informationAboutFounders.founderForeign
      LET founderForeign_size = length(foundersForeign)
      LET founderMunicipalties = doc2.informationAboutFounders.founderMunicipalty
      LET founderMunicipalty_size = length(founderMunicipalties)
      //counting share per kind of owner
      //we chek there is at least one founder owner
      filter founder_size > 0
      LET founder_share = (
         for founder in founders
         LET share_size  = founder.founderShare.sizeOfShares.sizeOfSharePercentage
         filter share_size != null
         COLLECT commited_founder = share_size
         RETURN commited_founder
      )
      LET sum_founder_share = SUM(founder_share)
```

```
FILTER (sum_founder_share < 50 and sum_founder_share >=0)
//giving that company owner have a value less than 50,
//and bigger that 0, then we try ti find an undetermined value of shares
//that means a company with declared shares less than 100 %.
//counting shares of individuals
LET individual_share = (
FILTER founderIndividuals != null
    for individual in founderIndividuals
    LET share_individual  = individual.founderShare.sizeOfShares.
    sizeOfSharePercentage
    filter share_individual != null
    COLLECT commited_individual = share_individual
    RETURN commited_individual
 )
LET sum_individual_share = SUM(individual_share)
//counting shares of investment found
LET InvestmentFund_share = (
FILTER InvestmentFunds != null
for InvestmentFund in InvestmentFunds
   LET share_InvestmentFund  = InvestmentFund.founderShare.sizeOfShares.
   sizeOfSharePercentage
   filter share_InvestmentFund != null
   COLLECT commited_InvestmentFund = share_InvestmentFund
   RETURN commited_InvestmentFund
)
LET sum_InvestmentFund_share = SUM(InvestmentFund_share)
//counting shares of foundersForeign
LET foundersForeign_share = (
FILTER foundersForeign != null
for founderForeign in foundersForeign
   LET share_founderForeign  = founderForeign.founderShare.sizeOfShares.
   sizeOfSharePercentage
   filter share_founderForeign != null
   COLLECT commited_foundersForeign = share_founderForeign
   RETURN commited_foundersForeign
)
LET sum_foundersForeign_share = SUM(foundersForeign_share)
//counting shares of municipality
LET founderMunicipalties_share = (
FILTER founderMunicipalties != null
for founderMunicipalty in founderMunicipalties
   LET share_founderMunicipalty  = founderMunicipalty.founderShare.sizeOfShares.
   sizeOfSharePercentage
   filter share_founderMunicipalty != null
   COLLECT commited_founderMunicipalty = share_founderMunicipalty
   RETURN commited_founderMunicipalty
)
LET sum_founderMunicipalties_share = SUM(founderMunicipalties_share)
```

```
        LET total = sum_founder_share+
        sum_individual_share+
        sum_InvestmentFund_share+
        sum_foundersForeign_share+
        sum_founderMunicipalties_share
        FILTER total == 0
        LET terminated = doc2.informationAboutTermination
        //check it is not previously sanctioned
        LET ofac_status_array =
        (
        FOR doc4 IN ofac_ownership_relation//previous relation link
           LET doc4id = doc4.owned_businessRegisterNumber
           FILTER businessRegisterNumber == doc4id
           LET ofac_status = doc4.ofac_santioned
           RETURN ofac_status//ofac_commited_share//length
        )
        LET status_ofac = ofac_status_array[0]
        filter status_ofac != null//current owner must be sanctioned
        filter sanction_value == null//current company must not be sanctioned
        UPDATE doc1 WITH {ofac_santioned: "wcs2"} IN ofac_ownership_relation//
        LIMIT 10
RETURN    {status_ofac, total }//
```

13. **set_wcs_03**

```
//set ofac wcs_03
FOR doc1 IN ofac_ownership_relation
   LET doc1id = doc1._to//identifier of owned
   LET ofac_ownerID1 = doc1.owner_businessRegisterNumber//identifier of owner
   LET ofac_ownedID1 = doc1.owned_businessRegisterNumber//identifier of owner
   LET sanction_value = doc1.ofac_santioned//checking if there is previous sanction flag
      FOR doc2 in egrul_data_collection
      LET doc2id = doc2._id
      FILTER doc1id == doc2id
      LET businessRegisterNumber = doc2.businessRegisterNumber.literal
      //with the company identify, we proceed to coount ownerships if any
      LET founders = doc2.informationAboutFounders.founder
      LET founder_size = length(founders)
      LET founderIndividuals = doc2.informationAboutFounders.founderIndividual
      //owned id as businessRegisterNumber
      LET founderIndividual_size = length(founderIndividuals)
      LET InvestmentFunds = doc2.informationAboutFounders.unitInvestmentFund
      //owned id as businessRegisterNumber
      LET InvestmentFund_size = length(InvestmentFunds)
      LET foundersForeign  = doc2.informationAboutFounders.founderForeign
      LET founderForeign_size = length(foundersForeign)
      LET founderMunicipalties = doc2.informationAboutFounders.founderMunicipalty
      LET founderMunicipalty_size = length(founderMunicipalties)
      //counting share per kind of owner
```

```
//we chek there is at least one founder owner
filter founder_size > 0
LET founder_share = (
   for founder in founders
   LET share_size  = founder.founderShare.sizeOfShares.sizeOfSharePercentage
   filter share_size != null
   COLLECT commited_founder = share_size
   RETURN commited_founder
)
LET sum_founder_share = SUM(founder_share)
FILTER (sum_founder_share < 50 and sum_founder_share >=0)
//giving that company owner have a value less than 50,
// and bigger that 0, then we try ti find an undetermined value of shares
//that means a company with declared shares less than 100 %.
//counting shares of individuals
LET individual_share = (
FILTER founderIndividuals != null
for individual in founderIndividuals
   LET share_individual  = individual.founderShare.sizeOfShares.
   sizeOfSharePercentage
   filter share_individual != null
   COLLECT commited_individual = share_individual
   RETURN commited_individual
)
LET sum_individual_share = SUM(individual_share)
//counting shares of investment found
LET InvestmentFund_share = (
FILTER InvestmentFunds != null
for InvestmentFund in InvestmentFunds
   LET share_InvestmentFund  = InvestmentFund.founderShare.sizeOfShares.
   sizeOfSharePercentage
   filter share_InvestmentFund != null
   COLLECT commited_InvestmentFund = share_InvestmentFund
RETURN commited_InvestmentFund
)
LET sum_InvestmentFund_share = SUM(InvestmentFund_share)
//counting shares of foundersForeign
LET foundersForeign_share = (
FILTER foundersForeign != null
for founderForeign in foundersForeign
   LET share_founderForeign  = founderForeign.founderShare.sizeOfShares.
   sizeOfSharePercentage
   filter share_founderForeign != null
   COLLECT commited_foundersForeign = share_founderForeign
   RETURN commited_foundersForeign
)
LET sum_foundersForeign_share = SUM(foundersForeign_share)
//counting shares of municipality
```

```
      LET founderMunicipalties_share = (
      FILTER founderMunicipalties != null
      for founderMunicipalty in founderMunicipalties
         LET share_founderMunicipalty  = founderMunicipalty.founderShare.sizeOfShares.
         sizeOfSharePercentage
         filter share_founderMunicipalty != null
         COLLECT commited_founderMunicipalty = share_founderMunicipalty
         RETURN commited_founderMunicipalty
      )
      LET sum_founderMunicipalties_share = SUM(founderMunicipalties_share)
      LET total = sum_founder_share+
      sum_individual_share+
      sum_InvestmentFund_share+
      sum_foundersForeign_share+
      sum_founderMunicipalties_share
      LET diff = 100 - total
      LET wcs3 = sum_founder_share + diff
      FILTER (total < 100 and wcs3 > 50)
      LET terminated = doc2.informationAboutTermination
      FILTER terminated == null//if terminated, nothing else to do
      //check it is not previously sanctioned
      LET ofac_status_array =
      (
      FOR doc4 IN ring2_ofac_ownership_relation//previous relation link
         LET doc4id = doc4.owned_businessRegisterNumber
         FILTER businessRegisterNumber == doc4id
         LET ofac_status = doc4.ofac_santioned
         RETURN ofac_status
      )
      LET status_ofac = ofac_status_array[0]
      FILTER status_ofac != null
      FILTER sanction_value == null
UPDATE doc1 WITH {ofac_santioned: "wcs3"} IN ofac_ownership_relation//
LIMIT 10
COLLECT WITH COUNT INTO length
RETURN    length//
```

Appendix B

Ownership Output

Collection	Total Ownership
Iteration 1	875
Iteration 2	883
Iteration 3	339
Iteration 4	215
Iteration 5	275
Iteration 6	237
Iteration 7	281
Iteration 8	205
Iteration 9	149
Iteration 10	204
Iteration 11	199
Iteration 12	156
Iteration 13	239
Iteration 14	187
Iteration 15	151
Iteration 16	209
Iteration 17	192
Iteration 18	150
Iteration 19	235
Iteration 20	181
Iteration 21	151
Iteration 22	224
Iteration 23	210
Iteration 24	148
Iteration 25	234

Appendix C

OFAC Santioned Ownerships

Collection	**OFAC** 100 %	**OFAC** 50 %	**OFAC** AGG
Iteration 1	117	15	260
Iteration 2	100	62	17
Iteration 3	17	16	3
Iteration 4	4	2	0
Iteration 5	0	0	0

Appendix D

Worst Case Scenario Results

Collection	WCS_1	WCS_2	WCS_3
Iteration 1	115/**42**	178/**136**	10
Iteration 2	50/**17**	42	0
Iteration 3	2/**1**	15/**14**	0
Iteration 4	0	0	0

Acronyms

AGG Aggregation. 13

AQL ArangoDB Query Language. 12, 16, 17

EGRUL Russia Federal Tax Registry. 4, 5

OFAC Office of Foreign Assets Control. 4, 7, 12, 13, 15, 17, 32

SDN Specially Designated Nationals And Blocked Persons List. 4

USA United States of America. 4

WCS Worst Case Scenario. 15–18, 33