DEGREE PROJECT IN COMPUTER SCIENCE AND ENGINEERING, SECOND CYCLE, 30 CREDITS

STOCKHOLM, SWEDEN 2019

Finding patterns in procurements and tenders using a graph database

MICHAEL SWORDS

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Finding patterns in procurements and tenders using a graph database

MICHAEL SWORDS

Master in Computer Science
Date: December 16, 2019
Supervisor: Håkan Lane
Examiner: Elena Troubitsyna
School of Electrical Engineering and Computer Science
Host company: Tendium
Swedish title: Hitta samband i upphandlingar och anbud med en grafdatabas

Abstract

Graph databases are becoming more and more prominent as a result of the increasing amount of connected data. Storing data in a graph database allows for greater insight into the relationships between the data and not just the data itself.

An area that has a large focus on relationship is the area of public procurements. Relationships such as who created which procurement and who was the winner. The procurement data today can be very unstructured or inaccessible which means that there is a low amount of analysis available in the area. To make it easier to analyse the procurement market there is a need for a proficient way of storing the data.

This thesis provides a proof of concept of the combination of public procurements and graph databases. A comparison is made between two models of different granularity, measuring both query speed and storage size. There has also been an exploration of what interesting patterns that can be extrapolated from the public procurement data using centrality and community detection.

The result of the model comparison shows a distinct increase in query speed at the cost of storage size. The result of the exploration is several examples of interesting patterns retrieved using a graph database with public procurement data, which show the potential of graph databases.

Sammanfattning

Grafdatabaser blir mer och mer populära till följd av ökningen av väldigt sammankopplad data. Lagring av data i en grafdatabas möjliggör större insikt i förhållanden mellan data och inte bara uppgifterna i sig.

Ett område som har fokus på relationer är offentliga upphandlingar. Relationer såsom vem som skapade en upphandling och vem som vann den. Det finns för närvarande ingen bästa praxis och ingen lättillgänglig analys i området. För att göra det enklare att analysera upphandlingsmarknaden behöver vi ett effektivt sätt att lagra uppgifterna.

Det här arbetet har gjort ett koncepttest av hur man kan kombinera offentliga upphandlingar och grafdatabaser. Två databasmodeller med olika granularitet har jämförts, gällade frågehastighet och lagringsstorlek. Examensarbetet har även gjort en undersökning av vilka intressanta mönster som går att extrapolera från upphandlingsdata med hjälp av grafalgoritmer som detekterar centralitet och gemenskap mellan noder.

Resultaten från modelljämförelsen visar en tydlig ökning av frågehastighet till kostnad av lagringsstorlek. Resultaten från utforskningen av sambanden är flera exempel på intressant extraktioner av mönster från en grafdatabas med offentlig upphandlings data.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Definitions . . . 2
- 1.3 Research Question . . . 3
- 1.4 Contribution . . . 3
- 1.5 Limitations . . . 4

2 Background **5**
- 2.1 Tenders and Procurements . . . 5
- 2.2 Graph Databases . . . 6
  - 2.2.1 General information . . . 6
  - 2.2.2 Graph Database Comparisons . . . 7
  - 2.2.3 Neo4j . . . 9
- 2.3 Graph Analysis . . . 12
  - 2.3.1 Centrality detection . . . 12
  - 2.3.2 Community detection . . . 14
- 2.4 Related Work . . . 17
  - 2.4.1 Assessing the potential for detecting collusion in Swedish public procurement . . . 18
  - 2.4.2 Information Extraction using a Graph database . . . 18
  - 2.4.3 Risk mitigation using graph centrality . . . 19

3 Method **21**
- 3.1 Data . . . 21
  - 3.1.1 Tenders electronic daily (TED) . . . 21
  - 3.1.2 Parsing the TED data . . . 23
  - 3.1.3 Company data . . . 23
- 3.2 Graph database and System specification . . . 24
- 3.3 Setting, Libraries and Plugins . . . 24

3.4 Graph Models . . . 25
3.4.1 Model 1 - Simple . . . 25
3.4.2 Model 2 - Extended . . . 28
3.5 Migration . . . 32
3.6 Speed and Size . . . 33
3.6.1 Query execution time . . . 33
3.6.2 Retrieving Store sizes . . . 34
3.7 Graph analysis . . . 34
3.7.1 Projections of the graphs . . . 35
3.7.2 Centrality and Communities . . . 35

4 Results **38**
4.1 Graph model comparisons . . . 38
4.1.1 Query execution time . . . 38
4.1.2 Store sizes . . . 39
4.2 Communities . . . 41
4.2.1 Entity to Entity . . . 41
4.2.2 SNI to CPV . . . 44
4.2.3 Cosine similarity of Entities based on CPV codes . . . 48
4.3 Centrality . . . 51
4.3.1 Entity to Entity . . . 51
4.3.2 Entity to NUTS County . . . 53
4.3.3 CPV to NUTS County . . . 54

5 Discussion **56**
5.1 Data quality . . . 56
5.2 Graph model comparisons . . . 56
5.3 Graph analysis . . . 57
5.4 Example Use Cases . . . 58
5.5 Threats to validity . . . 59
5.6 Sustainability . . . 59

6 Conclusions **60**
6.1 Future work . . . 61

Bibliography **62**

A Ted XML structure **66**

Chapter 1

Introduction

1.1 Motivation

The theory of using graph structures in databases is not new. In the 1970s Edgar. F. Codd proposed a relational model which is very similar to today's graph databases [1]. But the interest in Graph databases has only recently escalated with the rise of huge social media companies and the enormous amounts of connected data generated [2]. Storing data in a graph database allows for greater insight into the relationships between the data, and not just the data itself.

An area that has a large focus on relationships is the area of public procurements. Relationships of interest such as who created which procurement and who was the winner, which procurements are geographically similar and which are within the same trade.

Public procurements have an estimated value of ~€2 Trillion in Europe [3]. The value of Swedish procurements alone is roughly 683 Billion SEK which is close to 17% of Sweden's annual GDP [4]. Today the data for procurements are legally open for the public but a lot of the information about procurements is inaccessible or incomplete as there are no best practices and no real analysis available. This has resulted in problems such as very few bidders, expensive administration work and irrelevant requirements for the procurements. To make it easier to analyse the procurement market a more proficient way is needed for storing the data. A graph database would likely be a good candidate for this, given the amount of different relationships procurement data

has.

This thesis looks to explore the usage of graph databases in the area of public procurements, by creating a proof of concept. For the exploration of the combination of graph databases and public procurements, two questions have been asked. One focusing on the differences in data modelling by comparing query speed and storage size. The other, what findings that are possible to be extrapolated from the data using the native graph structure.

1.2 Definitions

Here follows a few definitions of how certain words are used in this thesis

Agency Governmental Agency

Company Company that have won procurements

Entity Either an Agency or a Company

Procurement A Process where a state, county or municipality is looking to make a purchase of goods, services or construction. In this thesis we are only working with winning tenders as a result of missing data.

Procurer Entity that has created a procurement

SNI Swedish Standard Industrial Classification (SNI) is used to classify enterprises and workplaces according to the activity carried out [5].

CPV Common procurement vocabulary (CPV) used by procurers to describe the type of supplies, works or services of the contract. It has a Tree structure comprising of codes up to 8 digits (plus a check digit, which serves to verify the 8 digits).

- The first two digits identify the divisions
- The first three digits identify the groups
- The first four digits identify the classes (sub group)
- The first five digits identify the categories
- the last three digits identify a finer description of the categories (sub categories)

[6]

NUTS Nomenclature of Territorial Units for Statistics (NUTS). Similarly to CPV, the NUTS codes also have a hierarchical structure. In Sweden there are three levels:

1. Nuts 1 - Three lands: East, south and north.
2. Nuts 2 - Eight National Areas, such as: East Middle Sweden, Middle Norrland and Småland and the islands
3. Nuts 3 - 21 Counties, such as: Stockholm county, Skåne county and Gävleborg County.

TED Tenders Electronic Daily (TED) is the online version of the 'Supplement to the Official Journal' of the EU, dedicated to European public procurement. All public tenders above a specific contract value must be published in the Supplement of the Official Journal of the European Union.

1.3 Research Question

- What is an appropriate model for procurements and tenders based on query speed and storage size? Which type of nodes should exists and what should their properties be. What are the relevant relationships between the nodes?
- What are some interesting patterns and relationships that can be found using a graph database e.g. can we find patterns such as largest influencers or clustering/communities of companies or procurers?

1.4 Contribution

The goal of this thesis is to explore possibilities when storing procurement data in a natively connected format. This goal contributes to organizations working with procurements who wish to get a new outlook on the area as it shows what information that is possible to retrieve. It also contributes to the field of graph databases as it shows another usage of the technology in an area that has not yet been explored. As this is a proof of concept that shows the potential of graph analysis it can be built upon by others with different graph algorithms and data sets to find other patterns that this thesis has not explored.

1.5 Limitations

The data used in this thesis is limited to only Swedish public procurements and involves procurement descriptions, procurers and winning companies. Information on public procurements was collected from relatively structured data sources providing general information about completed procurements.

Actual procurement notices and contracts contain more detailed information than what is provided in established structured data sources, but since extracting such data in a scalable way is very difficult, no such information has been used in this thesis. The resulting graph database is a proof of concept and is not of production standard.

Chapter 2

Background

2.1 Tenders and Procurements

Public procurements is a process where generally a state, county or municipality is looking to make a purchase of goods, services or construction. In order to prevent fraud, waste or corruption there are several laws and regulations. A tender gives a bid where it provides the required information that is requested from the procurement.

Some interesting Swedish statistics about procurements from the Swedish agency for public procurement [4]: The value of public procurements in Sweden 2016 was 683 billion which is about 17% of Sweden's BNP. With 18 525 total procurements in 2017 69% of them were from municipalities and 39% of all procurements were construction work. There were on average 4.1 tenders per procurement. About half of the procurements was from micro or small companies.

There is currently no Swedish agency that gathers data about Swedish public procurements. Compared to many of the EU's members there is no national declaration or announcement of public procurements. Instead there is a private market for announcement-databases, Visma Commerce AB [7] being one of the biggest actors on the Swedish market. According to EU directives if a procurement is over a certain threshold it is announced in Tenders Electronic Daily (TED) which is EU's shared advertisement database. The thresholds depend on the type of contract e.g. public works is 5 548 000 EUR, Service contracts and Supplies contracts are 221 000 EUR [8]. Even though a pro-

curement is below the threshold it is seen as good practice to announce your procurement in the TED database. The data in TED has however not been verified or quality tested by the Swedish procurement agency.

2.2 Graph Databases

2.2.1 General information

A graph is a set of nodes with edges connecting them. In graph databases the node can be viewed as an objects and an edge some relationship between two objects. Graphs are useful in representing a wide variety of datasets such as social networks, molecules or power grids.

The most popular graph model for graph databases is the labelled-property graph in which both the nodes and edges can store properties represented by key/value pairs. They can also be labelled to be assigned a category. For example, in figure 2.1 where Sven and Walter is part of the label *User* and the relationship between them is labelled *Knows*.

The properties in this graph are the names, i.e. both user and food nodes has a property called *name*. Relationships can also have properties e.g. in the edge from Sven to Walter there exists a property *Since*.

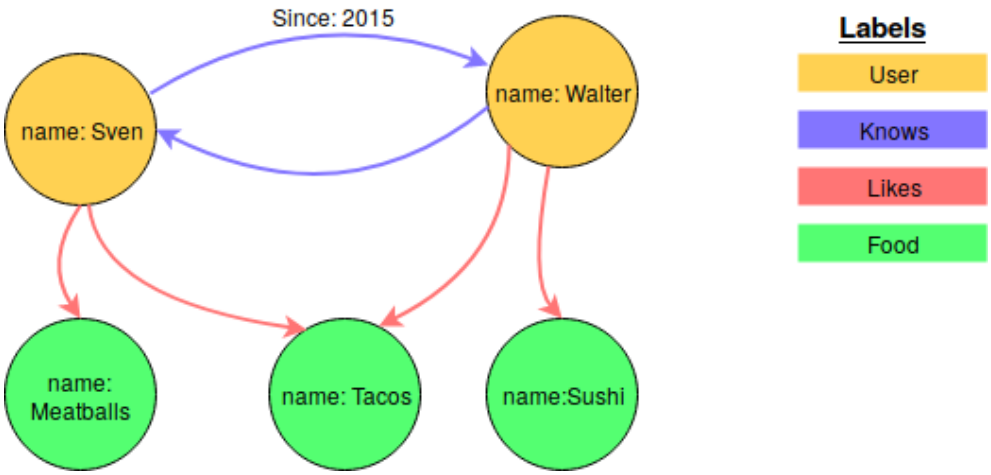


Figure 2.1: Small graph with properties and labels

A graph database has native processing capabilities if it has index-free adja-

cency, meaning that the connected nodes directly point to each other in the database as opposed to relational databases where join tables are often used to connect information. The underlying storage of graph databases varies, some use non-native storage such as relational databases or object-oriented databases where they store the graph data in tables. The native graph databases use NoSQL structures such as key-value store or document-oriented databases.

Using index-free adjacency gives a great advantage over traditional relational databases when querying using joins, as the index-free adjacency means that the joins are already precomputed and stored as relationships. Index-free adjacency does however sacrifice the efficiency of queries that do not use joins.

In order to apply CRUD (Create, Read, Update or Delete) operations on graph databases you cannot use the traditional SQL language. As there are many different developers of graph databases, several query languages have been introduced such as Gremlin, Cypher, SparQL and GraphQL. To make the learning curve less steep most of these languages are inspired by SQL as most developers are familiar with SQL.

2.2.2 Graph Database Comparisons

There has been several articles and papers that compare different graph databases. The two shown below were used as a guide to choose which graph database to use in this thesis.

HPC scalable graph analysis benchmark

In one of the articles a graph database performance comparison was done using the HPC scalable graph analysis benchmark [9]. This is a compact application that consists of four "kernels" which accesses a data structure representing a weighted, directed graph. The kernels are:

- Kernel 1, Measuring the edge and node insertion performance
- Kernel 2, Measuring the time needed to find a set of edges that meet a condition.
- Kernel 3, Measuring the time spent building sub-graphs in the neighborhood of a node

- Kernel 4, Estimating the traversal performance of the whole graph. (Traversed edges per second)

They compared four native graph databases: Neo4j [10], Jena [11], HypergraphDB[12] and Sparksee[13] (DEX). They found that Sparksee and Neo4j were the most efficient graph database implementations trading places at the top of the performance tests [9].

Graph database benchmark GDB

In an article by Salim Jouili and Valentine Vansteenberge [14] they recognized that the HPC benchmark does not analyze the effect of additional concurrent clients. So, they developed another benchmarking framework named GDB (Graph database benchmark). This tool can be used to simulate real graph database workloads with any number of concurrent clients performing any type of operation on any type of graph. A user defines a benchmark that first contains a list of databases to compare and then a series of operations which is to be run on each database. The benchmark is then executed using a module whose responsibility is to start the databases and measure the time required to perform the operations specified in the benchmark. Lastly a statistic module gathers and aggregates the measurements and produces a report. The operations performed on the databases were:

- Load workload - Starting a database and load it with a particular dataset
- Traversal workload - Finding all the nodes that are a certain number of jumps away from a node
- Intensive workload - Running a number of parallel clients that concurrently sends basic request to the graph database.

They tested four databases Neo4j, DEX, Titan[15] and OrientDB[16]. Travesal and intensive workloads had graph sizes with 250,000 and 500,000 nodes. In their load workload results the graph databases increase in time rather linearly except Neo4j which after being the fastest up to a certain point jumps up in computation time which they suspected to be due to a daemon related to data reorganization on disk. The traversal workload favored Neo4j and was TitanCassandra's downfall. In the Intensive workload it was hard to determine which database got the best results. Overall Neo4j and DEX were the top performers

2.2.3 Neo4j

Neo4j was first released in 2010 and is currently one of the most popular graph databases [17]. The popularity ranking was based on the frequency of searches, number of mentions in search engines and questions on sites such as Stack Overflow.

Neo4j stores the data in several "store" files which each contain a specific part of the graph e.g. nodes, relationships, labels and properties [18]. The node store file is a fixed-size record store which means that it can find a node in O(1) time if we know it's ID. A node record contains a flag if the record is in use, pointers to lists of relationships/edges, labels and properties. The relationship record is also of fixed-size, it contains a pointer to it's type, pointers to the next and previous relationship records for each of the start and end nodes, and a flag indicating whether it is the first in a relationship chain.

Cypher

Cypher is Neo4j's own developed query language for their graph database [18]. Cypher is designed to intuitively describe graphs using diagrams, i.e queries that asks the database to find data matching a specific pattern. For example:

```
MATCH (p:Person) -[:likes]-> (f:Food)
RETURN p
```

A node is depicted by two parentheses e.g. *(p)*. In order to specify a specific type of node, the language uses labels such as *(p:Person)* where *p* is a variable and *Person* is a label. To represent edges in the query you use arrows –> between two nodes. These edges are directional, but it is also possible to query using an undirected edge. You can also depict nodes and edges without variables names e.g *()*, *(:Person)* or *-[:likes]->* as shown in the query above. The MATCH keyword is similar to the Select in SQL where we state that we want to read/select something and RETURN specifies what kind of results you want to get out of the query. As you saw in the figure 2.1 edges can also have labels and variables.

Procedures and Transactions

Procedures in Neo4j are mechanisms that perform operations on the database, User-defined procedures and functions that allows the user to write custom code which can be called directly from the Cypher language and is usually the way to extend functionality. The database comes with a number of built-in procedures and functions such as listing all constraints on the database, listing all property keys or simply listing all procedures.

All database operations that access the graph, indexes, or the schema must be performed in a transaction. Neo4j supports the ACID property:

- Atomicity - guarantees that either every statement in a transaction is executed or if one fails then no changes are made.
- Consistency - guarantees that the database's state is valid, before and after a transaction. i.e. consistent.
- Isolation - guarantees that multiple transactions can occur concurrently as if they were executed sequentially.
- Durability - guarantees that once a transaction has been committed it can always recover the results of the transaction.

Transactions acquire locks from nodes and edges when they are created, deleted or when editing properties of the nodes or edges. The locks are only released when the transaction is done. It is also possible to explicitly obtain a lock in a Cypher query using the $\_lock$ statement. There are three events points that can be used as triggers, before committing a transaction, after committing a transaction and after a rollback of a transaction has been done.

Constraints and Indexes

Similarly, to standard databases, Neo4j allows for indexes that enhance performance in search and full-text search. But it is also possible to apply constraints to a node type's attribute. Constraints can be seen as the primary key of a node, allowing no other node of the same type to have the same attribute value.

Graph modeling

The books *graph databases 2nd edition* [18] and *learning neo4j* [19] introduces some best practices when graph modeling. As modeling is hugely dependent on the area, which questions that are asked and which datasets that are used. The best practices from the two books are more pointers and tips rather than strict guidelines.

- Illustrate the design - Graph data model is whiteboard friendly, what you see on paper translates very easily if not exactly into the database.
- Use Cases - Try to pinpoint what information we want from the data; which queries are going to be asked. From those queries identify objects and relationships between the objects.
- Granulate - Firstly. how granular should the graph be, what should be properties and what should be their own nodes. This depends a lot on the previous point of use cases and the queries that are going to be asked to the database. Secondly, we should also try and label edges and nodes to be distinct enough to allow specific path traversal but broad enough to be applied to similar associations.
- N-ary Relationships - It can be useful to identify relationships that are associated to more than two things, an intuitive improvement is to create a new node that links all the things together.

Some pitfalls to avoid:

- Nodes representing multiple concepts - Trying to have one type of node represent many different things.
- Unconnected graphs - Having unconnected nodes in the graph gives nothing but struggles, as the database uses traversal to find nodes you lose out on a lot of query power.
- Nodes into relationships - Do not try to make objects into relationships, making a object into a relationship removes the possibility to associate other objects to that relationship. If an relationships has a lot of attribute-like properties and duplicate values in different relationships it is possible that the relationship should be remodeled to a node instead.

2.3 Graph Analysis

2.3.1 Centrality detection

Centrality detection refers to the process of identifying the most important nodes in a graph or the nodes that have the most influence in the graph. There has been a lot of research in the area of centrality and this section covers some of the measures used to evaluate a node's importance/influence.

Degree centrality

Degree centrality is the simplest of the centrality measures, it is defined as the number of ingoing and outgoing edges from a node. Degree centrality was introduced by Linton C. Freeman 1979 in his article Centrality in Social Networks: Conceptual Clarification [20]. In a directed graph there are two degrees to consider, indegree and outdegree. Where indegree is the number of edges pointing to the node and outdegree is the number of edges going outwards. The average degree of a network is the total number of edges divided by the total number of nodes. The degree distribution is the probability that a randomly selected node will have a certain number of edges [21]

$$Degree(x) = deg(x) \tag{2.1}$$

Closeness centrality

The closeness centrality of a node, measure of how fast the node is able to spread information through a graph. The intuition being that the most central node is the node with the smallest distance to all other nodes. To calculate a node's closeness, we take the sum of its shortest paths to all other nodes and invert it.

$$Closeness(x) = \frac{1}{\sum_y d(x,y)} \tag{2.2}$$

where: x is the node, n is the number of nodes in the graph and $d(x,y)$ is the shortest distance between node x and y.

It is common to normalize this score so that it represents the average length of the shortest paths rather than their sum. This adjustment allows comparisons of the closeness centrality of nodes of graphs of different sizes. The equation for normalized closeness centrality is:

$$NormalizedCloseness(x) = \frac{n-1}{\sum_y d(x,y)} \tag{2.3}$$

Betweenness centrality

Introduced by Linton Freeman, Betweenness Centrality is a measure of how many times a node acts as a bridge along the shortest path between two other nodes [22].

$$Betweeness(x) = \sum_{s \neq x \neq t} \frac{p(x)}{p} \tag{2.4}$$

where: x is the node, p is the total number of shortest paths between nodes s and t. p(x) is the number of shortest paths between nodes s and t that pass through node x

Betweenness Centrality is very interesting as it is not always the nodes with the highest visibility that are the most important, sometimes the most relevant nodes are the ones that connect large components of the graph with only a few number of edges.

Eigenvector centrality and Pagerank

Eigenvector centrality is a measure of the influence of a node based how influential its neighbours are. In a graph G with an adjacency matrix A the eigenvector centrality of a node x is defined as:

$$x_v = \frac{1}{\lambda} \sum_{t \in G)} a_{v,t} x_t \tag{2.5}$$

which can be rewritten as the eigenvector equation:

$$Ax = \lambda x \tag{2.6}$$

A variant of eigenvector centrality and one of the most known centrality algorithms is Pagerank which is used to rank web pages and was created by

Google's cofounder Larry Page. The algorithm measures the transitive influence of nodes, i.e. how likely it is to traverse to a specific node from other nodes. The assumption is that a more important website is more likely to be pointed to by other webpages.

In order to measure the influence of a node we look at the quantity and quality of incoming edges to a node and then create an estimation of how important that node is. To stop dead ends i.e. nodes that have no outgoing nodes you usually implement "random teleportation" which can be thought of as edges that go from that node to every other node with a very low edge weight. The equation presented in the original paper is [23]:

$$Pagerank(x) = (1-d) + d(\frac{Pagerank(T_1)}{C(T_1)} + ... + \frac{Pagerank(T_n)}{C(T_n)}) \quad (2.7)$$

where: a node x has edges from nodes T_1 to T_n, d is a damping factor which is set between 0 and 1. It is usually set to 0.85. $1-d$ is the probability that a node is reached directly without following any edge. $C(T_n)$ is defined as the out-degree of a node T.

The standard Pagerank algorithm is iterative and will run on the graph until either the scores have converged or a max of iterations have been reached. This is known as the Power Method [24] where we are computing the principal eigenvector by starting with the uniform distribution of the pagerank values.

2.3.2 Community detection

Community detection is the process of finding groups of nodes in graphs that are highly connected, these groups are called communities. Show below are some different approaches to community detection.

Strongly connected components - Tarjan's algorithm

Using strongly connected components is one of the earlier concepts for finding communities in graphs [18]. A directed graph is called strongly connected if there exists a path between any two pair of nodes. For example in figure 2.2 the sub-graph {A, B, E} is a strongly connected component.

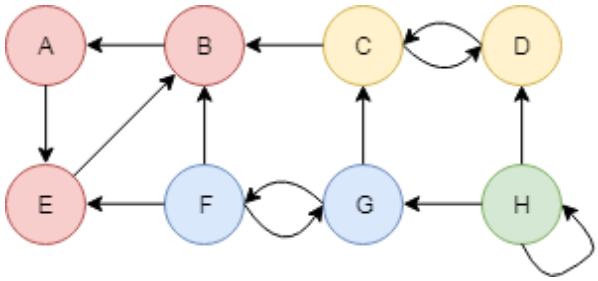


Figure 2.2: Strongly connected components

There are several algorithms that compute strongly connected components in linear time. One of which is Tarjan's algorithm, which was published in 1972 by Robert Tarjan [25]. The algorithm is a depth first search, where we maintain a stack of nodes that have been visited. Nodes are added to the stack if they are explored for the first time and are removed when a complete component has been found. Each node has a low-link value which is the lowest node id reachable from that node. When we visit a node in the depth first search, we give it an id, a low-link value, mark it as visited and add it to the stack. When the depth first search reaches a point where we cannot go any further then we look at the node we just checked and if it is on the stack we take the min of the low-link values between the node we just checked and the previous node to propagate the low-link values backwards. When we have visited all neighbours in a node and its low-link values equals its id we pop all nodes with that low-link value as we have now discovered a strongly connected component.

Markov Cluster algorithm

The Markov clustering algorithm also known as the MCL algorithm was invented by Stijn van Dongen and was presented in the thesis Graph clustering by flow simulation 2000 [26]. The algorithm is as follows, given a graph we create a column stochastic matrix. Which is a matrix where each column sums to 1 and an element in the matrix is the probability that from x:th node we move to y:th node. The rows in the matrix would be a nodes in-bounding edges with their corresponding probabilities. After that we do the following steps.

1. Input is a Graph, inflation parameter r.
2. Create the transition matrix with the probabilities.
3. Expand the matrix $M = M * M$

4. Inflate the matrix, $M = M.^r$, element wise exponentiation.
5. Normalize the columns
6. Prune low values that are below a threshold.
7. Repeat steps 3-6 until we converge.
8. Interpret resulting matrix to discover clusters.

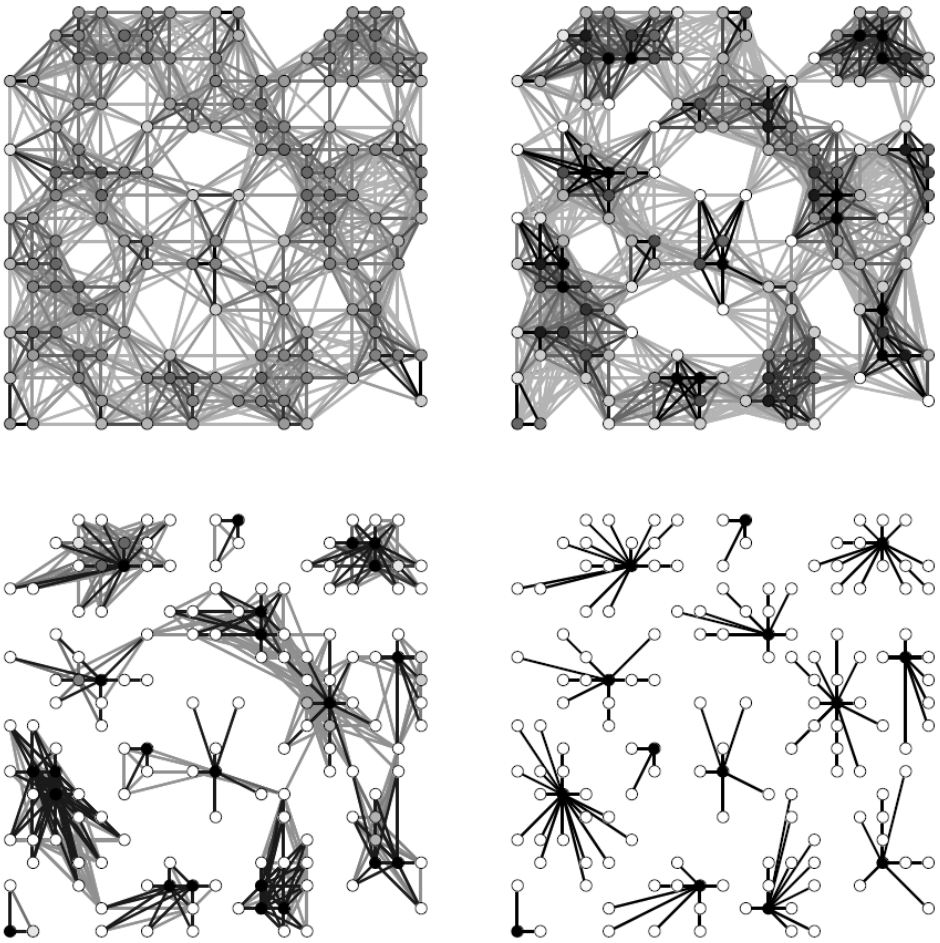


Figure 2.3: MCL convergence, top left to bottom right [26]

Nodes flowing into the same sink nodes are assigned the same cluster labels. Convergence is not proven in the thesis; however it is shown experimentally that it often does occur. Some limitations are that they output many small clusters [27], and that it does not scale well [28].

Louvain modularity

Modularity optimization is one of the most popular and widely used method for community detection in graphs [29]. Modularity is a value between -1 and 1 that measures the density of edges inside communities to edges outside communities, its equation is shown below (2.9) [30].

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j) \tag{2.8}$$

Where: A_{ij} is the weight of the edges between i and j, $k_i = \sum_j A_{ij}$ is the sum of the weights of the edges attached to node i, c_i is the cluster to which node i is assigned, the δ function $\delta(u, v)$ is 1 if u =v and 0 otherwise and $m = \frac{1}{2} \sum_j A_{ij}$.

Optimizing these values is practically impossible as there are too many iterations to go through. Instead heuristic algorithms are used. One of the better algorithms is the Louvain Method for Modularity optimization which is a method created by Blondel from the University of Louvain [30].

The method is divided into two phases that are repeated iteratively. The first phase is a greedy assignment of nodes to clusters by optimizing modularity locally, were each node checks which neighbour to join for best change of modularity. In the second phase it aggregates nodes belonging to the same community and builds a new graph whose nodes are the communities. Once the second phase is done creating the new graph the two phases are run again until the modularity stops improving.

It has shown to be accurate and one of the fastest of the community detection algorithms [31] [32].

2.4 Related Work

This section introduces some examples of what has been done with graph databases and research on the state of procurement data in Sweden

2.4.1 Assessing the potential for detecting collusion in Swedish public procurement

The Swedish Competition Authority gave Dr.Mihály Fazekas and Dr. Bence Tóth from Cambridge university a commission to analyse databases with data of public procurements in Sweden and the EU in order to find indicators of collusion [33]. The two databases are Visma Opic and Tender Electronics Daily (TED), a data quality assessment was made which showed that several vital pieces of information were lacking with a missing ratio of tens of percentages. E.g:

- Buyer's address.
- Award criteria - Specifies whether the contract is based on lowest price or price + quality.
- CPV code precision - Product code of a given tender.
- Contract length.
- Bidder's name, id, number of bidders
- Bid price, winning bid or not.

2.4.2 Information Extraction using a Graph database

In an article published by Thashen Padayachy, Brenda Scholtz and Janet Wesson: An Information Extraction Model Using A Graph Database To Recommend The Most Applied Case [34]. They presented a model to aid legal researchers in accessing the most applied case (MAC) for the field of law. Their model consists of four components, Information retrieval, Information extraction, graph database and query-independent ranking. A query from a user is parsed in the information retrieval module where a user gets a set of ranked results to select. After the user selects the ranked results the Information extraction module extracts the facts from the results that the user selected, which are then saved to the third component, the graph database. The fourth component returns a recommendation from the graph database consisting of the MAC to the user which is based on another ranking algorithm such as Pagerank [34].

The graph database allowed for increased performance due to the data being connected. When creating the graph, the authors bring up a good point of what

questions related to the data must be asked, in their case "What case is referred to in Case A?" or "How many cases does Case A refer to?". After a couple of iterations, they arrived at a graph shown in figure 1.1. Where edges are used to represent the decision made by a judge on a referred-to-case node.

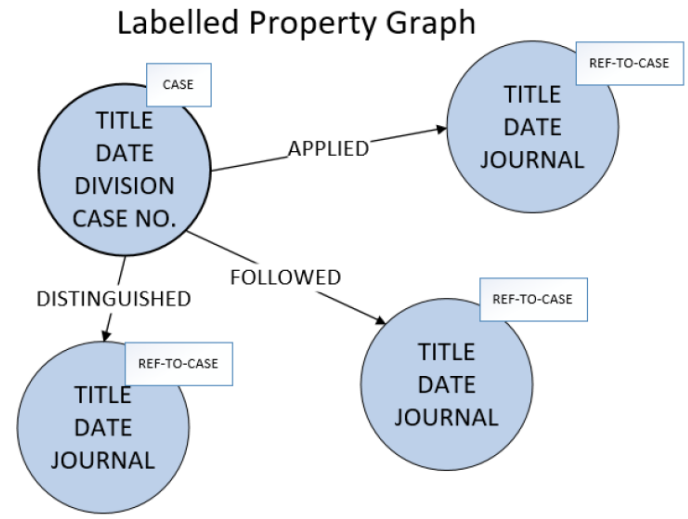


Figure 2.4: Labelled Property Graph [34]

2.4.3 Risk mitigation using graph centrality

Another use of graph databases and graph theory was done in the article: Risk mitigation strategies for critical infrastructures based on graph centrality analysis by George Stergiopoulosa, Panayiotis Kotzanikolaoub, Marianthi Theocharidouc and Dimitris Gritzalisa [35]. They used the centrality measures of degree, closeness, betweenness and eigenvector to evaluate the effectiveness of alternative risk mitigation strategies. The graphs they used are randomly generated with certain constraints that simulate common critical infrastructure dependency characteristics. The graph database Neo4j was used to implement the dependency analysis modeling tool. They bring up some interesting points about each centrality measure regarding effects of risk propagation.

Degree A high inbound degree centrality indicates a cascade resulting node and can be seen as sinkhole points of incoming dependency risk. The high outbounding nodes indicates a cascade initiating node where you might consider prioritizing mitigation controls.

Closeness With the shortest average distance to most nodes in the graph they are likely to be part of many dependency chains.

Betweenness Acting as a bridge to a high volume of nodes means that it is not initiating cascading failures, but there is still a high chance of contributing to multiple risk paths.

Eigenvector These nodes are interesting because they are connected to other nodes that also have high connectivity.

An experiment was done to demonstrate how often nodes simultaneously appear in the set of the most critical paths and sets of nodes with highest centrality values. Using varies constraints to simulate the risk in networks they randomly created a graph. Feature selection algorithms were used to detect correlations between high centrality metrics and high-risk nodes.

Their conclusions were that critical paths tend to involve nodes with high centrality values. An average of 74.4% of the most critical paths in the generated graphs contained a node with at least one high centrality metric. It was shown that a combination of the centrality measures yields a good strategy for risk mitigation, but that high closeness and degree centrality appear to have the highest impact on the overall risk.

Chapter 3

Method

3.1 Data

3.1.1 Tenders electronic daily (TED)

The procurements and tender data used was downloaded from TED which is the online version of the 'Supplement to the Official Journal' of the EU. TED publishes 520 thousand procurement notices a year, including 210 thousand calls for tenders which are worth approximately €420 billion. TED provides free access to public procurement notices from contracting authorities based in the European Union. The notices are published frequently each week. They are available on the website, in HTML, PDF and in XML format.

In this thesis we used the XML format. The full structure of the XML is shown in the appendix A. We have the root node containing the unique document number for TED, the unique official journal number and the document version. Below the root we have five different sections:

- Technical
- Links
- Translation
- Coded Data
- Form

The three sections of relevance for this thesis are the Technical, Coded data and Form sections.

Technical

The Technical section contains technical information of the original posters systems for internal management purpose of the production chain. Such as the identification of the notice in the original poster's production system, the date when the notice can be moved from on-line to archive in TED and list of languages codes in which the notice is published in full translation.

Coded data

The Coded Data section is divided in three groups of data.

- Ref OJS - Contains the data related to the publication.
- Notice Data - Contains data related to the notice.
- Codif data - Contains the META data (codification) related to the notice.

Form

The Form section is dependent on the document version referred to in the root element. The form's children are based on the form number ranging from F01 to F25 in document version R2.0.9.S01.E01 and F01 to F19, T01, T02 in document version R2.0.8.S02.E01. The form numbers are references to the different forms that can be submitted e.g. contract notice, notice on a buyer profile or Contract award notice which is the one of most interest and is more reliably found in the XMLs. The Contract award notice form contains five sections:

- Contracting authority - Information about the entity that created the procurement.
- Object - Information about the procurement itself.
- Procedure - How the procurement was carried through.
- Award contracts, one for each division of award - Information about the suppliers.

- Complementary info - Additional information.

The full structure is shown in appendix A.

3.1.2 Parsing the TED data

Parsing of the TED data was done by stepping through each XML file and getting the relevant fields from it. A requirement for a procurement being used was if it was possible to identify a procurer. The script saved the extracted data into several CSV files:

- Procurements - The relevant information about the procurement such as title, description, reference number.
- Buyers/Procurers - Information about the procurer such as name, organizational number, address and of course the procurement id that the buyer is associated with.
- Suppliers/Companies - Information about the company that won, containing similar information as the buyer.
- CPVs - Contains the CPV codes that were specified in the procurement and the procurement id.
- Contacts - Contact information found in the procurement. Such as name, email and the organization number that the contact is associated with.

The CSV files were stored in a S3 bucket which is Amazon's simple storage service for object storage to later on be migration into a graph database.

3.1.3 Company data

The company data was provided by Tendium and contained complementary information about companies. The information consisted of:

- General information such as: Name and organizational id, number of employees, registration and de-registration dates for companies, visiting and postal locations such as addresses, counties and municipals.
- SNI information with up to three codes and a title specifying different trades.
- People associated with a company, e.g. CEO, VD and Board members.

- Economical information such as Profit margin, Revenue and Operating profit.

3.2 Graph database and System specification

The databases were run in Amazon Web Services which is a cloud computing platform.

The graph database used was Neo4j as it was shown in the graph database comparison in section 2.2.2 that it performed very well. It is also currently one of the most popular graph database with a large community behind it.

The Neo4j databases were run on two EC2 instances of type M5a 2xlarge [36]. A M5a 2xlarge instance has eight 2.5 GHz AMD EPYC 7000 series processors and 32 GiB of Memory. What we are mainly after is the size of memory to allow for faster computation when working with the same data.

The storage used was an io1 SSD Provisioned IOPS (input/output operations per second) [37]. io1 SSDs are used for low-latency or high-throughput workloads which is advantageous as the database will frequently read from disk. The SSDs were of size 48 GiB with a max IOPS of 2000.

3.3 Setting, Libraries and Plugins

The Neo4j database version run was 3.5.8 enterprise edition in a docker container with 12 GiB max heap size, 12 GiB initial heap size and 12 GiB page-cache size. The following Neo4j Plugins/libraries were used:

- APOC (Awesome procedures on Cypher) [38] - A library for Neo4j that provides hundreds of procedures and functions [39].
- Graph algorithms [40] - A plugin/library which provides graph algorithms implementation that are parallelizable.

To visualize larger graphs, we first used Neoviz but it proved to only be able to handle up to 3000 nodes and edges, instead the tool Gephi [41] was used

as it can handle much larger graphs. Gephi also provides some implementations of the centrality and community algorithms specified in section 2.3.1 and 2.3.2.

3.4 Graph Models

To represent the data, two models were created. The models were created using the tips from the best practices described in section 2.2.3. The most influential tip was identifying the use cases of the graph database i.e which questions that would be queried to the database, which is a common approach as mentioned in the article about information extraction shown in section 2.4.2.

The questions of interest were: who created a specific procurement, which area of trade is the procurement part of and which company won. These questions are the most important as they allow us to look at the structure of the procurement market.

The differences between the two models were designed to measure the gained benefit in speed versus the increase in store size when having more relationships and nodes compared to having "larger" nodes containing more properties. Which can be interpreted as an analysis of the consequence of higher vs lower level of granularity in the models. The values that were deemed to be shareable by multiple nodes such as dates got their own nodes in the secondary model.

3.4.1 Model 1 - Simple

The first model's purpose is to be simple but be able to achieve what we want with our queries. The attributes of the different objects are stored inside the nodes and are shown in table 3.1 to 3.5. The Model is shown in figure 3.1 below.

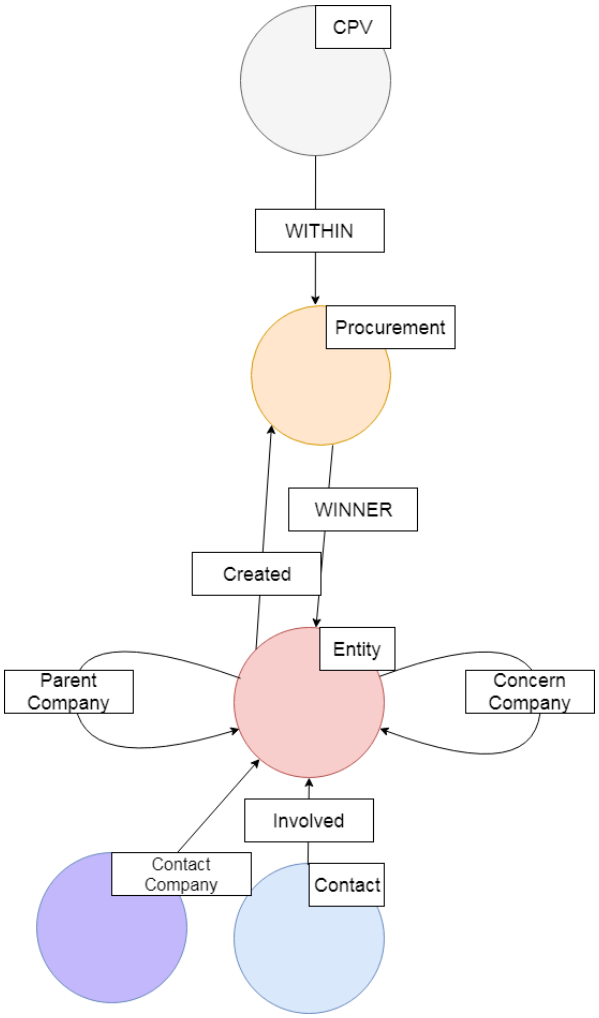


Figure 3.1: Model 1

The resulting graph has five different nodes:

The Entity node, which represents a Procurer or a Company. It can have possible relationships with Procurement, Entity, Contact and Contact Company nodes. The table 3.1 shows the properties in the node. The Visiting Info and Postal Info has been aggregated in the table, in reality the Address, County and Municipality etc. are in separate properties. Similarly, the SNI codes are in three different properties. The Red marked property represents the constraint (mentioned in section 2.2.3) on the node, similar to a primary key in a relational database.

Property	Description
Org id	Organisational number
Name	Name of the Procurer or Supplier
Visiting Info	Address, County, Municipal ..
Postal Info	Address, County, Municipal ..
SNI title	Description of main trade
SNI code 1-3	Codes representing different trades
Registration date	Date of registration

Table 3.1: Node: Entity

The Procurement node contains information about the procurement itself, such as the title, description and date dispatch. Properties shown in table 3.2

Property	Description
Title	Given title
Reference number	Given duplicate titles a reference number is needed.
Title Key	Combination of Title and reference number.
NUTS	Code that refers to a Region in Sweden
Description	Description
Date dispatch	Date when procurement was published in TED
Deletion date	Date when procurement is archived in TED

Table 3.2: Node: Procurement

The Contact node is the contact that was posted in the contract award notice by the Procurer and Supplier. The names were not always present in the data so the uniqueness of a contact was adapted to be name if it existed and otherwise email. Properties shown in table 3.3.

Property	Description
Primary key	Unique description of Contact. If name is found, we use the name otherwise we use mail
Name	Name of the Contact
Email	Email of the Contact
Number	Telephone number

Table 3.3: Node: Contact

The CPV node contains the CPV codes and descriptions of the codes. Properties shown in table 3.4.

Property	Description
Code	Numerical code that describes the trade
Desc_en	Description of the trade in english
Desc_se	Description of the trade in swedish

Table 3.4: CPV

The last node is the Contacts gathered from the company data. The edge between the Contact Company node and Entity node has several different labels depending on the relationship type, such as VD, Vice VD, Chairman. etc.

Property	**Description**
Name	Name
Id	Identification number

Table 3.5: Contact Company Data

Indexed properties

A secondary simple model was created with the minor difference that the queried properties are indexed. Which means that instead of searching through the nodes and their properties we go through a B+Tree [42] which is more efficient.

3.4.2 Model 2 - Extended

The second model extracts properties from the nodes in model one into new nodes. A calendar is created from nodes where we have a sub-graph of year, month and day. The Visiting and Postal addresses are created as new nodes such as municipality, county and postal number. SNI codes and SNI title gets their own nodes. An overview of the model is shown in figure 3.2.

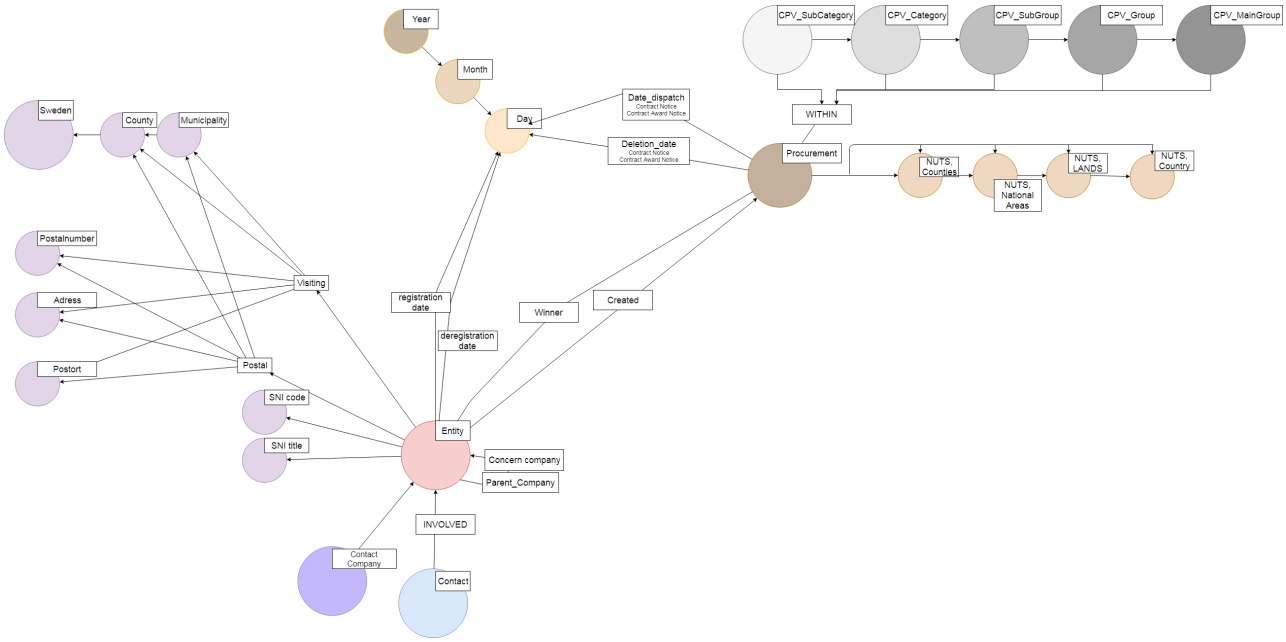


Figure 3.2: Model 2

In the Figure 3.3 below we can see the structure of the CPV and NUTS codes. CPV_MainGroup is the first description of a category and CPV_Group is within a CPV_MainGroup, CPV_SubGroup is within CPV_Group, etc. There is a uniqueness constraint on the Codes as no codes should be duplicate. Similarly, the NUTS codes are in a hierarchical structure with Country code being at the top followed by LANDS, National Areas and Counties. A procurement may have a relation to any of the nodes in the hierarchical structure of both CPVs and NUTS. The structure is shown in figure 3.3

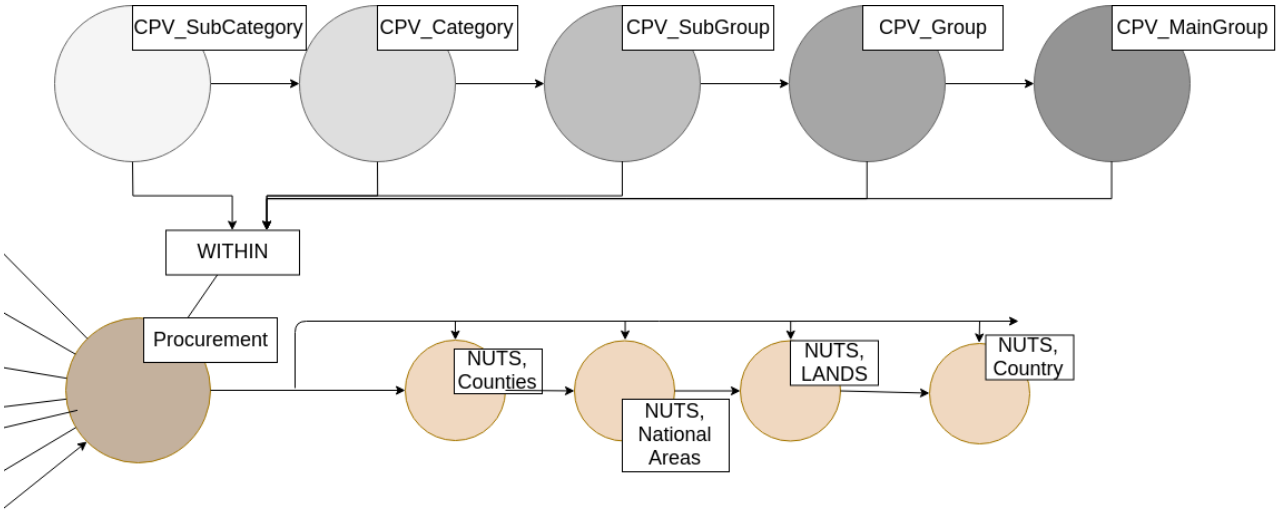


Figure 3.3: Model 2 CPV and NUTS

The calendar that was created to store dates using nodes of days, months and years is shown in table 3.4. Procurements and entities point to a specific day. That day points to a specific month and that month points to a specific year. This allows for quick traversal from one day to another day as we can get from one day to another day in just a few steps. We can also easily see what happened during a specific day as we only need to traverse the edges going from that day.

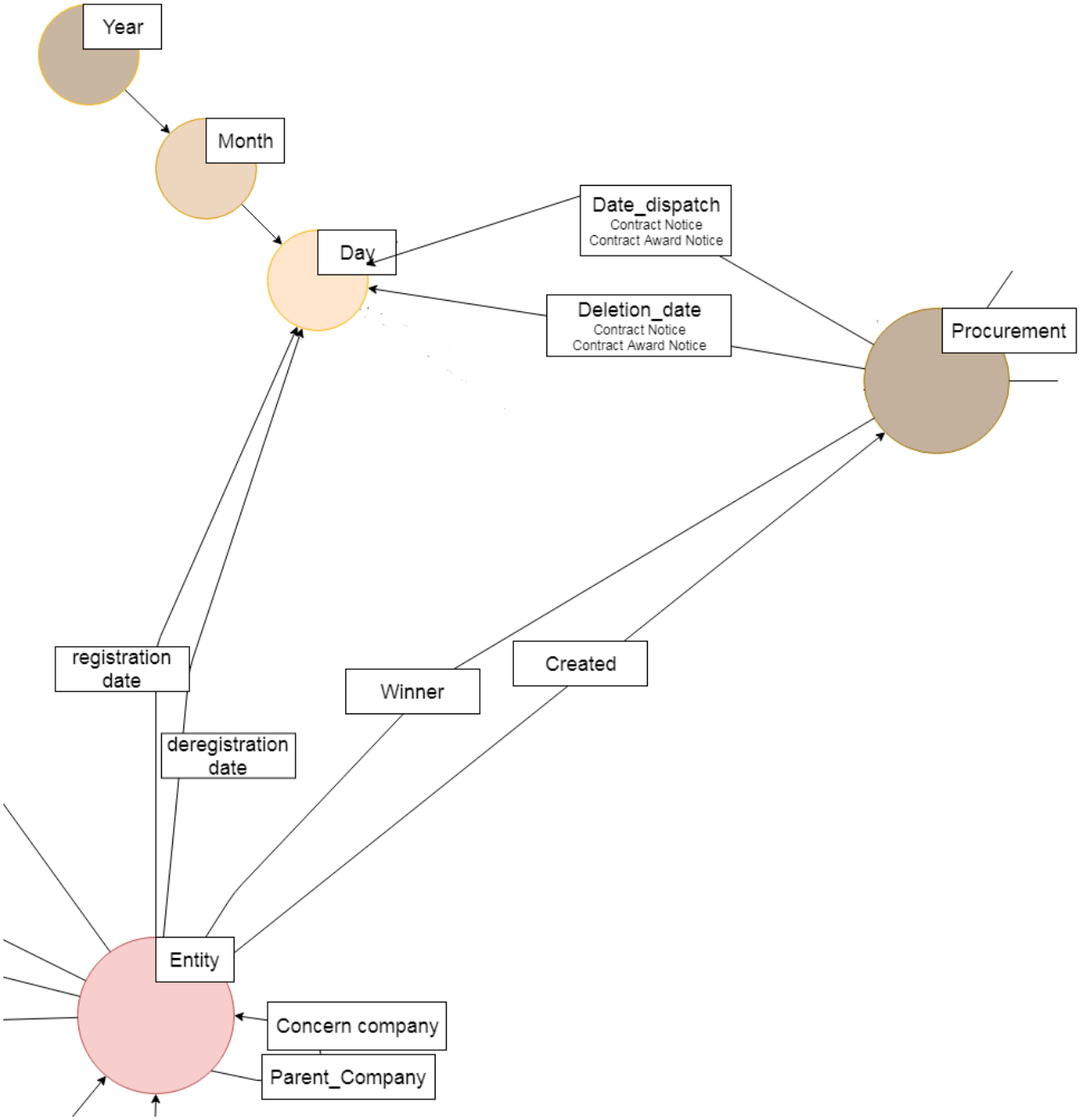


Figure 3.4: Model 2 Calendar

Some other properties that are shared by multiple Entities are the SNI codes and Visiting and Postal locations. Each Entity is related to several SNI codes and SNI titles, which can be used to find companies in the same trade. The

locations are stored as Counties, Municipalities, Postal number, etc. Which means we can quickly find entities located in the same area via either postal or visiting information.

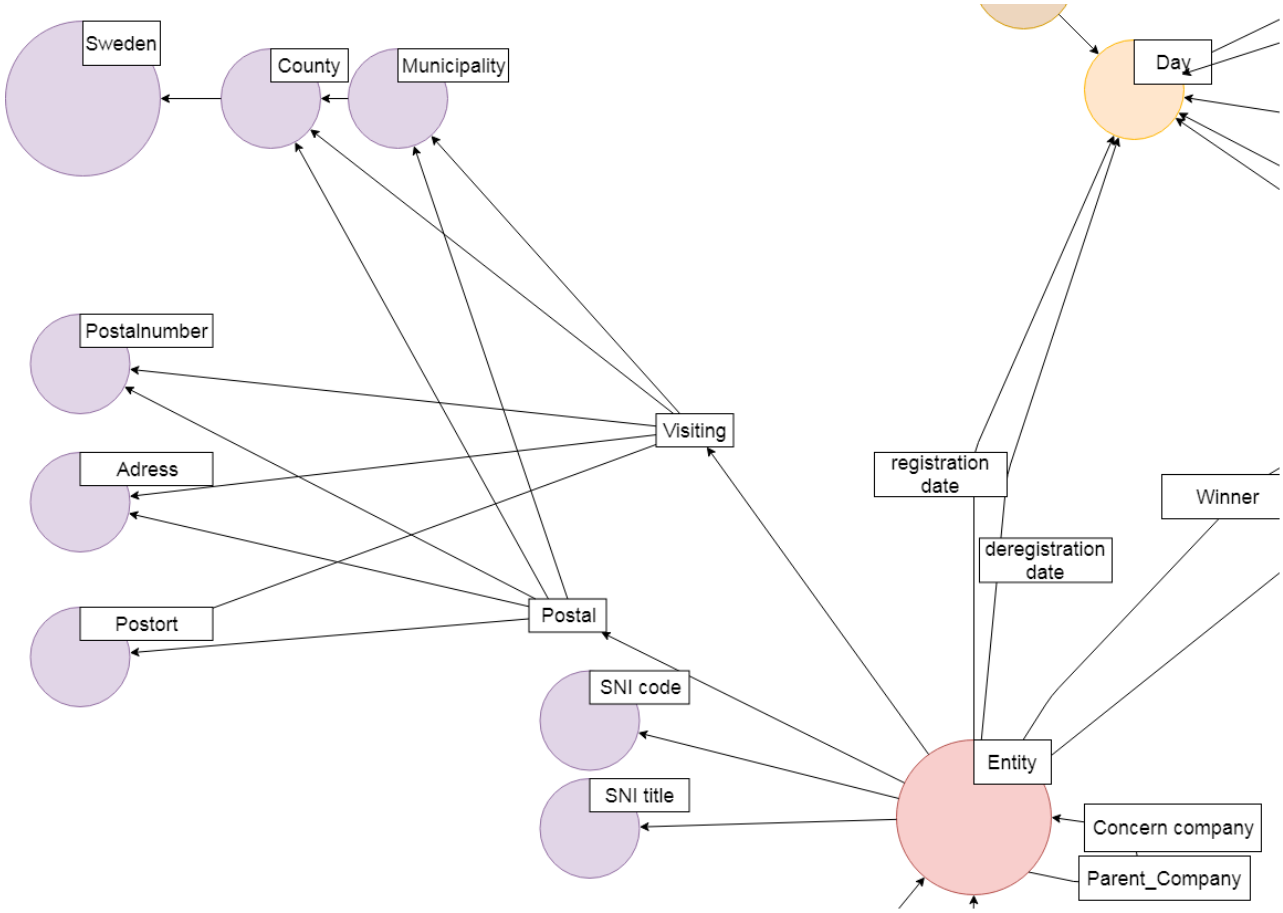


Figure 3.5: Model 2 Locations and SNI

3.5 Migration

With the models designed and data parsed and sent to the AWS S3 buckets it was possible to migrate the data into the two databases. The migration was done with cypher using Neo4j's Python Bolt driver [43] as shown in figure 3.6 below. The database reads the CSV files from S3 and creates the nodes, relationships and the appropriate properties in accordance to the cypher queries. Bolt is a network protocol developed by Neo4j.

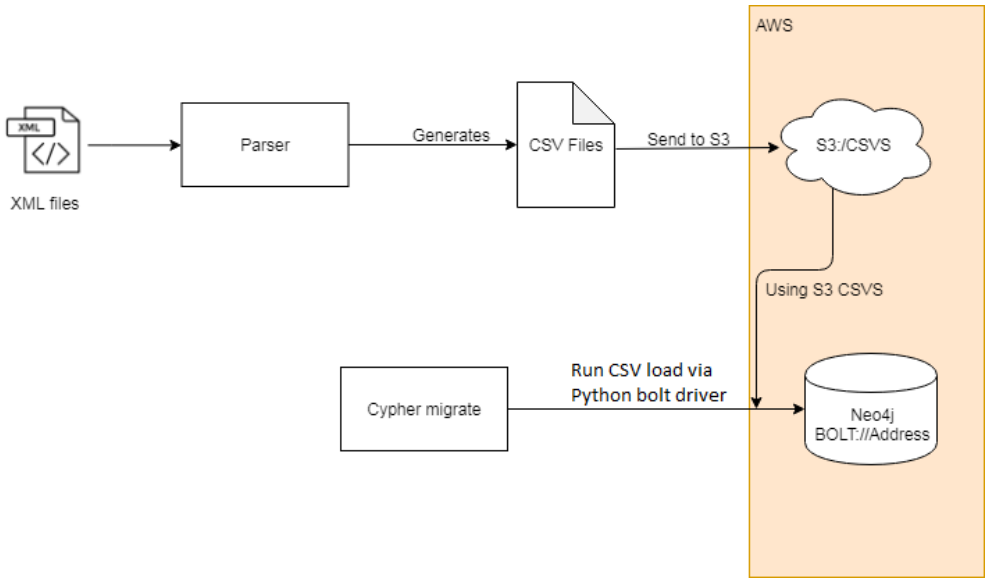


Figure 3.6: Parsing and Migration structure

3.6 Speed and Size

3.6.1 Query execution time

In order to compare models one 3.4.1 and two 3.4.2, two sets of five different queries were used, where each query is adapted to the model. The execution speed evaluated is Neo4j's response of available time and consume time. The available time is the time it took for the database to retrieve the first result and the consume time is the time it took for the database to consume the result. The estimation of the query speed is then the two values summed. The queries were run 100 times each giving an average and standard deviation. The queries were formulated for retrieving the following information from the two models:

1. How many entities created/won a procurement that were published on TED during a specific day.
2. How many entities created/won a procurement that were published on TED during a specific month.
3. How many entities created/won a procurement that were published on TED during a specific year.

4. How many entities are there in a specific municipality
5. How many entities are there in a specific county

The motivation behind these queries was that the information about dates and locations were stored in the entity and procurements nodes in model one 3.4.1 whereas in model two 3.4.2 they where in their own nodes. Here is an example of how query five was translated into the cypher language described in section 2.2.3 for the two models:

Example model 1:

```
MATCH (e:Entity) WHERE e.postal_county = 'Stockholms län'
OR e.visiting_county = 'Stockholms län'
RETURN count(distinct e)
```

Example model 2:

```
MATCH (c:County)-[:PostalCounty|VisitingCounty]-(e:Entity)
WHERE c.name = 'Stockholms län'
RETURN count (distinct e)
```

3.6.2 Retrieving Store sizes

The two models are setup up in two different databases running on two different EC2 instances. The store sizes were retrieved by using Neo4j's Python Bolt driver to call a JMX (Java Management Extensions) function that returns the size for each store. JMX is technology of the Java Platform and makes it possible to monitor or configure resources used by the system in this case Neo4j.

3.7 Graph analysis

To retrieve patterns that can be found in the representative graph of the procurement and company data, we examined the relationships and structural characteristics of the graphs. In the paper on risk mitigation using graph centrality [35] explained in section 2.4.3 the authors suggested that a combination of the centrality measures would yield the nodes with the most risk. Inspired by this, this thesis explored the three centrality measures: degree, closeness and eigenvector described in section 2.3.1 as combined they would likely give an accu-

rate centrality measure of the nodes. As modularity optimization described in section 2.3.2 is one of the most widely used methods to detect communities and the Louvain described in section 2.3.2 is accurate and arguably the fastest performing algorithm in this area it was used to detect communities.

3.7.1 Projections of the graphs

We are not always interested in the whole graph and instead want to focus on specific nodes and edges of a graph. This can be done with projections where we take out a sub graph. A way to do this is using the APOC library by creating virtual relationships or when running the graph algorithm procedures it is possible to specify which nodes and edges to include from the graph.

3.7.2 Centrality and Communities

Entity to Entity

Exploring the relationships between companies and procurers is one of the more interesting sub-graphs as it represents the public procurement market. To create this sub-graph we looked at all entities that have a procurement between them. For example, if procurer *e* sold to company *e2* we created a virtual relationship between them with a property value which is the amount of transactions that has been done between the two entities.

Here is what the cypher looks like.

```
MATCH q=(e:Entity)-[:Created]->(p:Procurement)-[:Winner]->
(e2:Entity)
WITH e,e2, count(q) AS counts
CALL apoc.create.vRelationship(e, 'Sold_to', {times: counts}, e2)
YIELD rel
RETURN e,e2, rel
```

The specified nodes and relationships were exported from Neo4j and imported into Gephi for visualization.

The centrality of the entities were calculated by applying the graph algorithm procedures described in section 2.3.1 using the Neo4j library of graph algorithm [40]. The communities of entities were identified using the Louvain algorithm described in section 2.3.2.

SNI to CPV

SNI codes and CPV codes are two different systems that both try to categorize areas of trade and products. Find a translation between the two systems would be helpful for many reasons such as when trying to identify companies that are potential bidders for a procurement or for getting a better understanding of what a code represents. One way to associate SNI codes to CPV codes is by creating the projection of companies' and procurers' SNI codes to the procurements' CPV codes. Similarly to the entity to entity method we can export the SNI to CPV projection to Gephi. The Louvain clustering was used to identify the clusters of SNI and CPV codes.

Entity similarity based on winning procurement types

To find companies that win similar procurements we make use of the procurement CPV codes. In this particular case we look at the CPV Group level, if a CPV code is below the level of Group we find its parent and traverse upwards until we find the associated Group level.

The second step is to create the virtual relationships from each company to each CPV Group code where the weight of the edge is the amount of times the company has won a procurement with that code.

The third step is to calculate the cosine similarity between each company based on the company's relationships to the CPV codes. If the similarity is high, we create a relationship called "*similar*" between the companies, i.e. we have decided that the companies are similar and show this by having a relationship between them. The cosine similarity similarity threshold was set to 0.95 to find companies that are very similar.

The sub-graph of interest is the set of company nodes and their similar relationships, and the final step is to take the projection of that graph. Using Louvain clustering we find which companies are in the same similarity community.

Entity to NUTS

The projection of entities that won procurements in specific NUTS regions is interesting because we can find how widespread companies and procurers

are geographically. By creating relationships from the entities to the NUTS regions they have won we can retrieve the Degree centrality of the companies and procurers, which gives us an indication of how national they are.

Chapter 4

Results

4.1 Graph model comparisons

4.1.1 Query execution time

The execution time is shown in bar Figure 4.1 where each set of three bars is one run query, from left to right: Entities from a specific county, Entities from a specific municipality, Procurements on a specific day, Procurements on a specific month, Procurements on a specific year. As shown in the figure the time for model one is noticeably higher than the indexed version and model two. The indexed version and model two are very similar. We can see that the only query, the day query, is fast for model one without indexing. but still not as fast as the indexed version and model two. The likely reason for the increased speed is that we only match one value and not a range of values or multiple values.

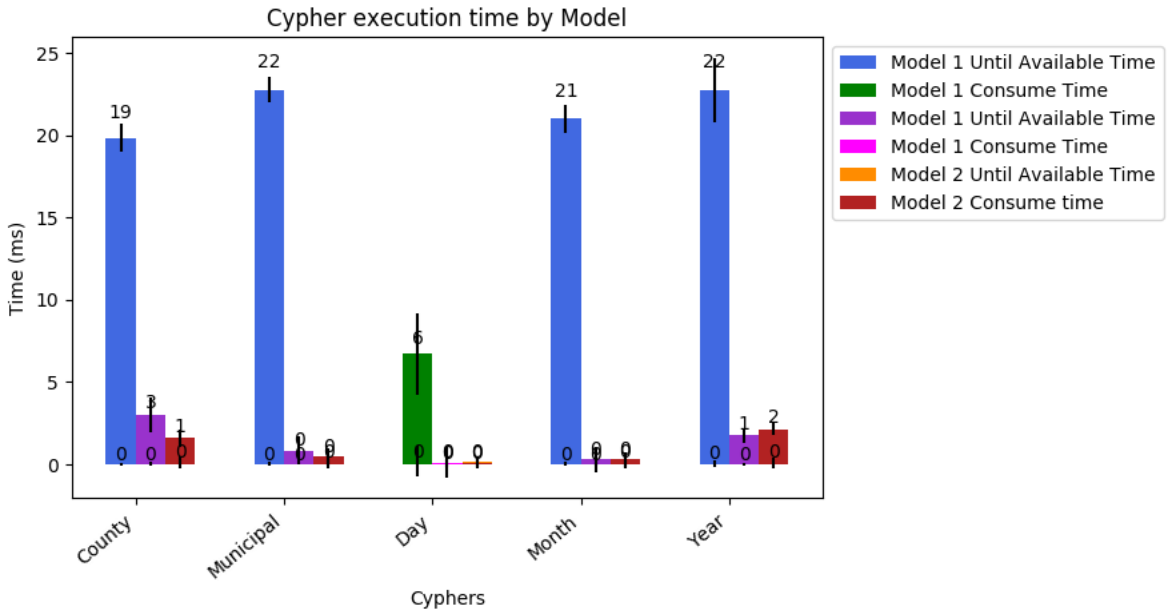


Figure 4.1: Cypher speeds

4.1.2 Store sizes

Shown below in figure 4.2 is the sizes of the stores for each model. As we can see the main differences are the considerable increase in Index Store size and Relationship Store. It is clear that the increase in relationships largely increases the size of the relationships store.

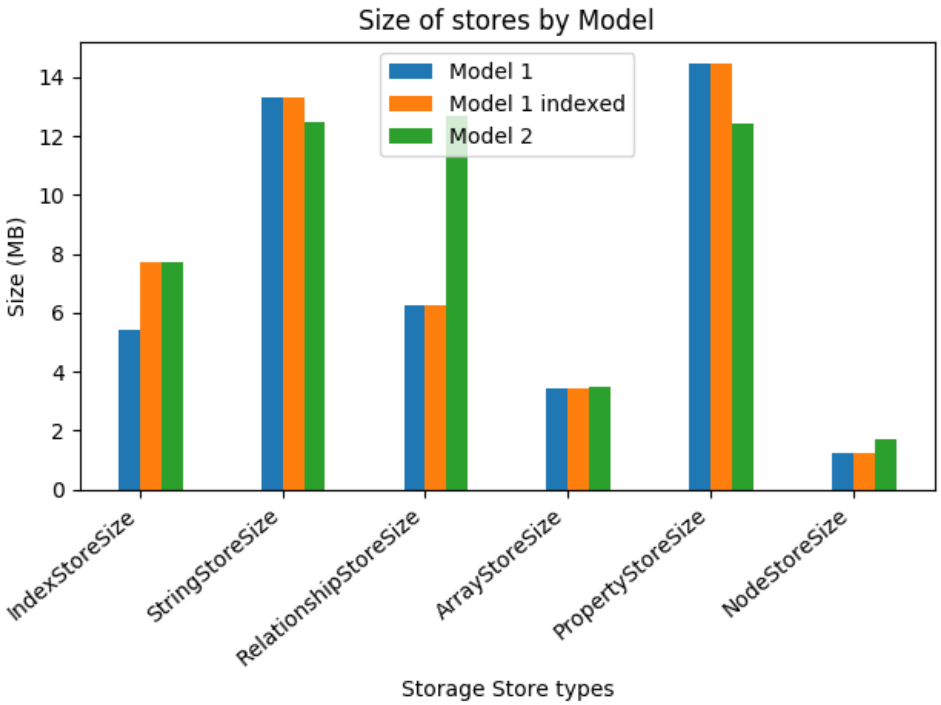
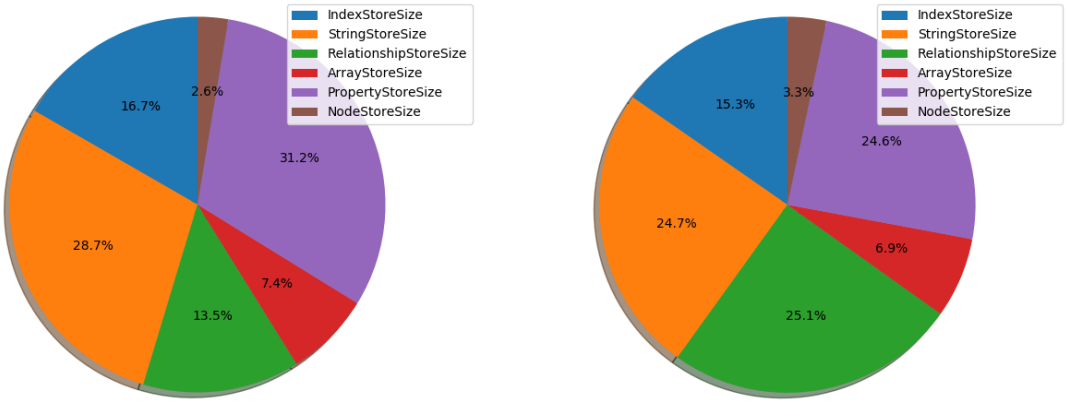


Figure 4.2: Store sizes models comparison

In the Pie charts in figure 4.3 below we can see the comparison of how comparatively large the stores are in the graph database. The relationship store size grows in twice the size in model 2.



(a) Model one store sizes

(b) Model two store sizes

Figure 4.3: Piecharts for store sizes

4.2 Communities

4.2.1 Entity to Entity

Using the projection of entity to entity from section 3.7.2 and community detection using the Louvain clustering we get the communities of companies and procurers. A visualization is shown in Figure 4.5. In the visualization each colour represents a cluster of Entities. We can see some well-formed clusters in e.g. the Brown, Orange and Dark Green parts of the graph. The modularity score of the clustering was 0.568.

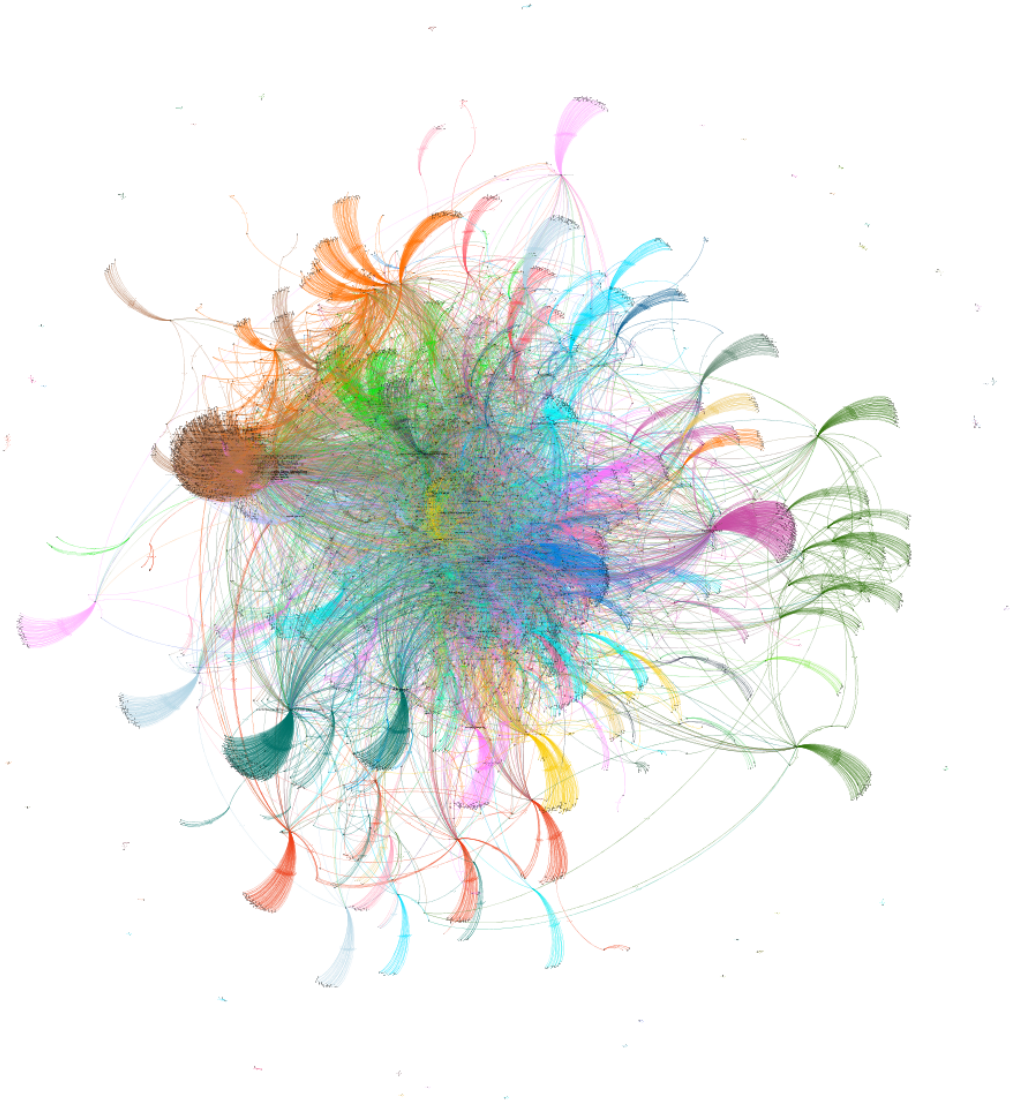


Figure 4.4: Entity to Entity, Louvain clustering

A zoomed in picture (Figure 4.6) of the Brown cluster shows that these are the companies working frequently with several county councils such as Stockholm, Skåne and Västernorrland. Since the county councils also are in the same cluster it is highly likely that the companies winning procurements from one council is also winning from others, thereby linking the county councils closer. There are also companies that have only won procurements from one agency which also gets adopted into the community of the agency.

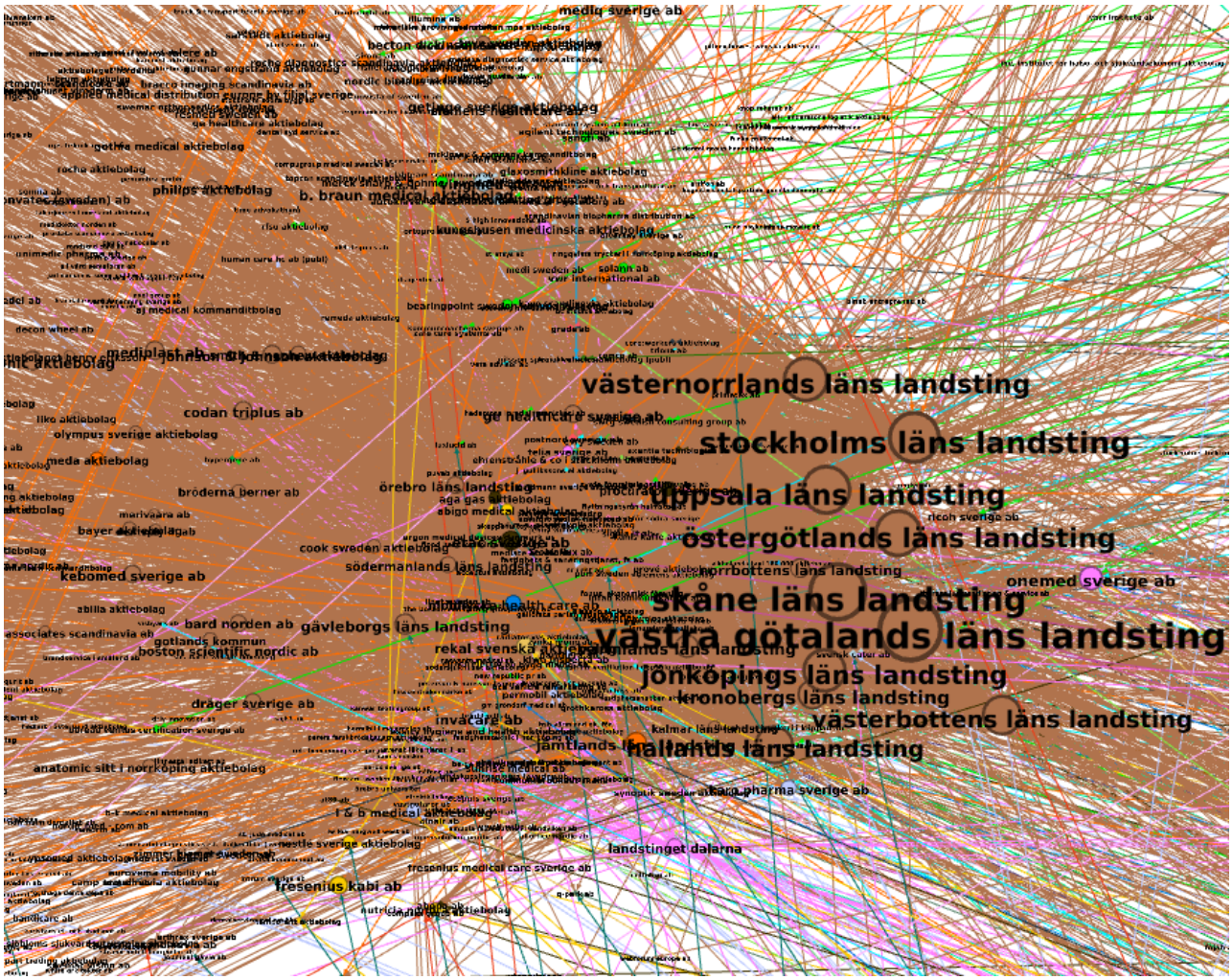


Figure 4.5: County cluster

Similarly, if we zoom in on the Orange cluster (Figure 4.7) we can see that the Universities has bundled together into a community, meaning the universities are creating procurements that are won by the same companies. We can see Lund's, Uppsala and Stockholm's university etc.

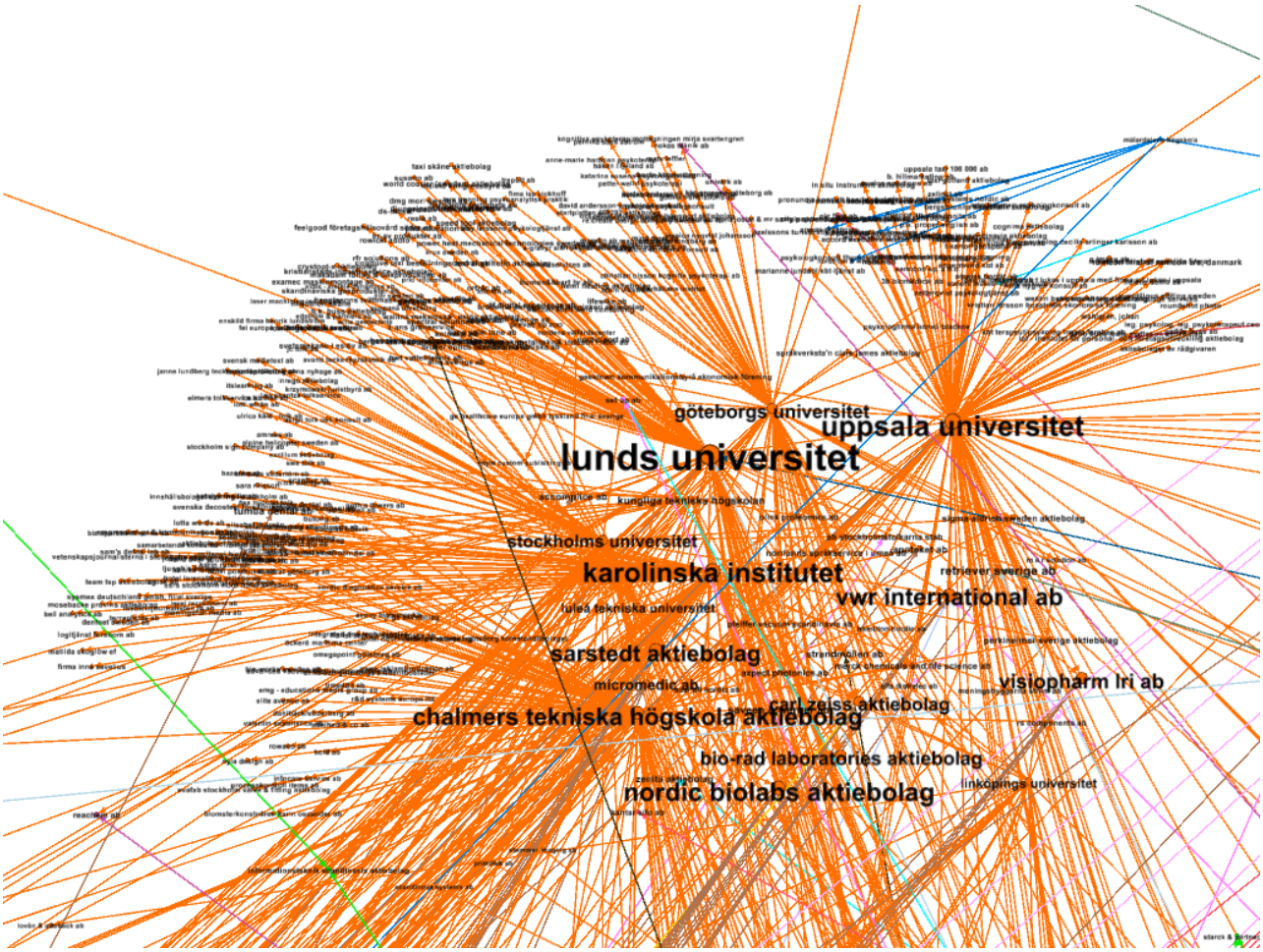


Figure 4.6: universities cluster

4.2.2 SNI to CPV

This section shows the results of using the projection of SNI to CPV Group, the edge weight is the number of times the two codes SNI and CPV have a relationship between them. The modularity score of the clustering was 0.736.

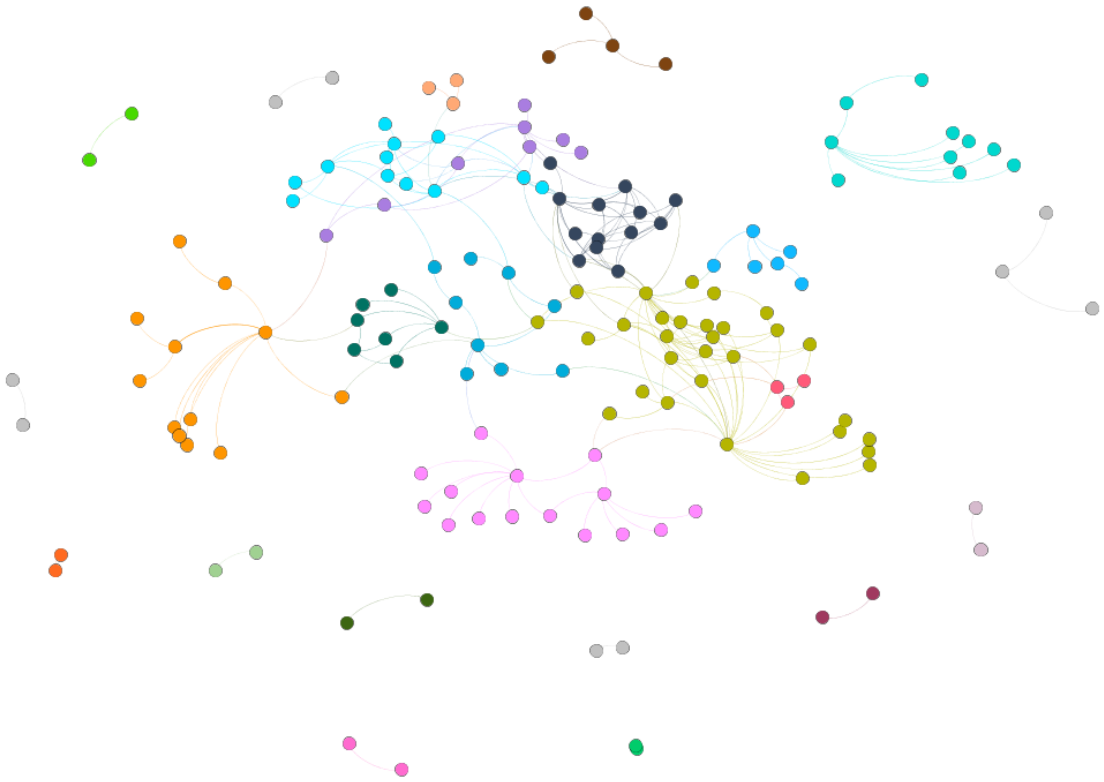


Figure 4.7: SNI to CPV. More than or equal to 100 occurrences, Louvain clustering

The clustering resulted in an orange cluster which groups Medical associated codes. We can see CPV codes such as health services, pharmaceutical product and Medical equipment and SNI codes such as production of medical and dental instrument, wholesale of medical equipment and apothecary goods, production of cycles and invalid-vehicles.

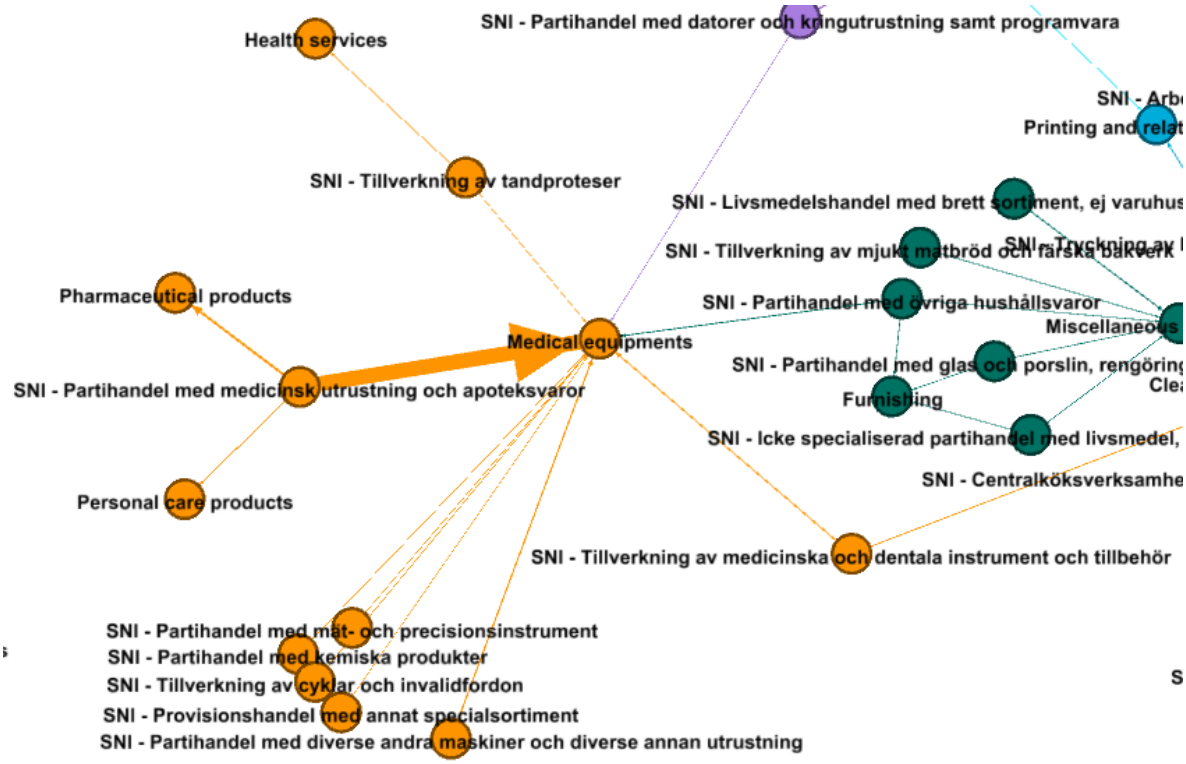


Figure 4.8: SNI to CPV. Medical

The light blue cluster below in figure 4.9 contains entertainment, marketing and media related CPV and SNI codes.

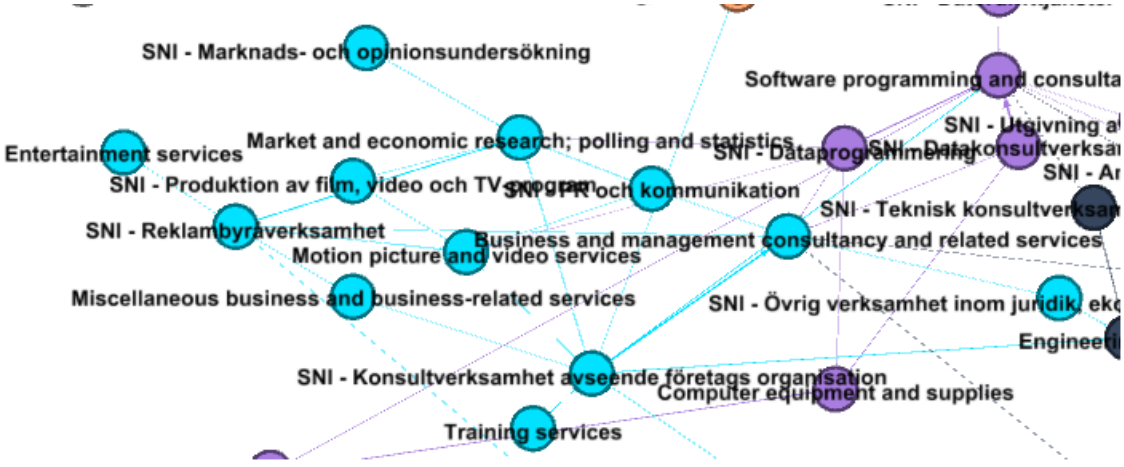


Figure 4.9: SNI to CPV. Media

The dark green cluster contains construction and installation of different sorts. For instance, CPV codes: Building installation work, Installation services of guidance and control system, works for complete or part construction and civil engineering work. SNI codes: Electrical installations, construction of roads and motorway, Heat and sanitize works.

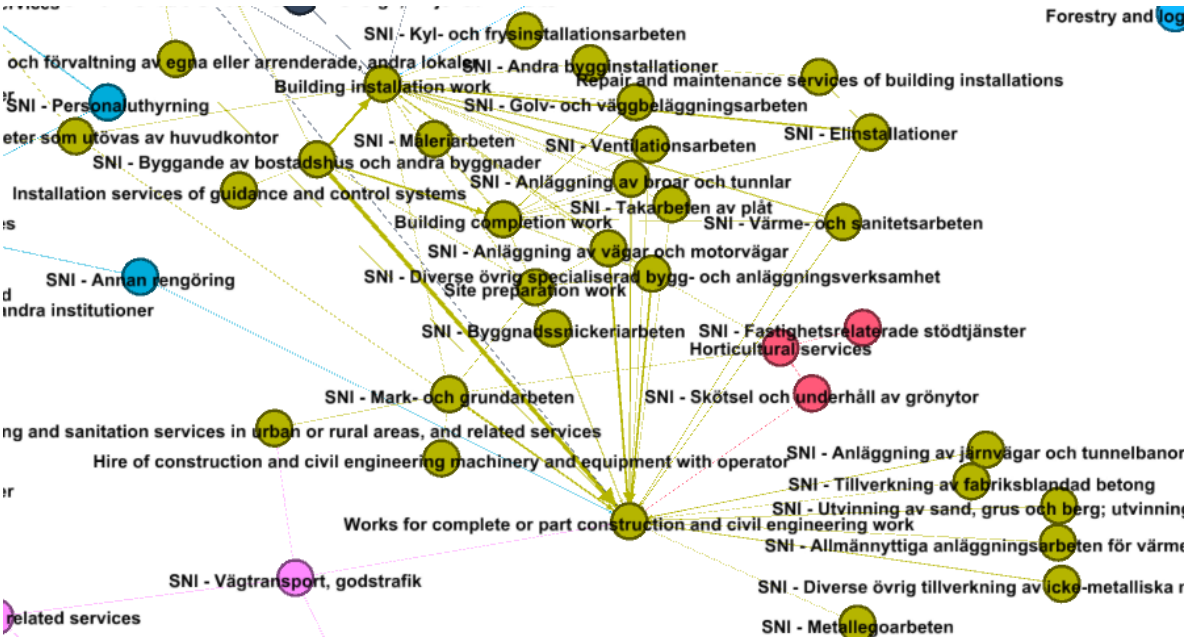


Figure 4.10: SNI to CPV. Construction and Installation

4.2.3 Cosine similarity of Entities based on CPV codes

This section shows the results of the steps done in the method section 3.7.2. Each colour is a cluster of Entities that are similar based on their won procurement's CPV codes. The clusters are really well formed which is reflected in the modularity score of 0.919.

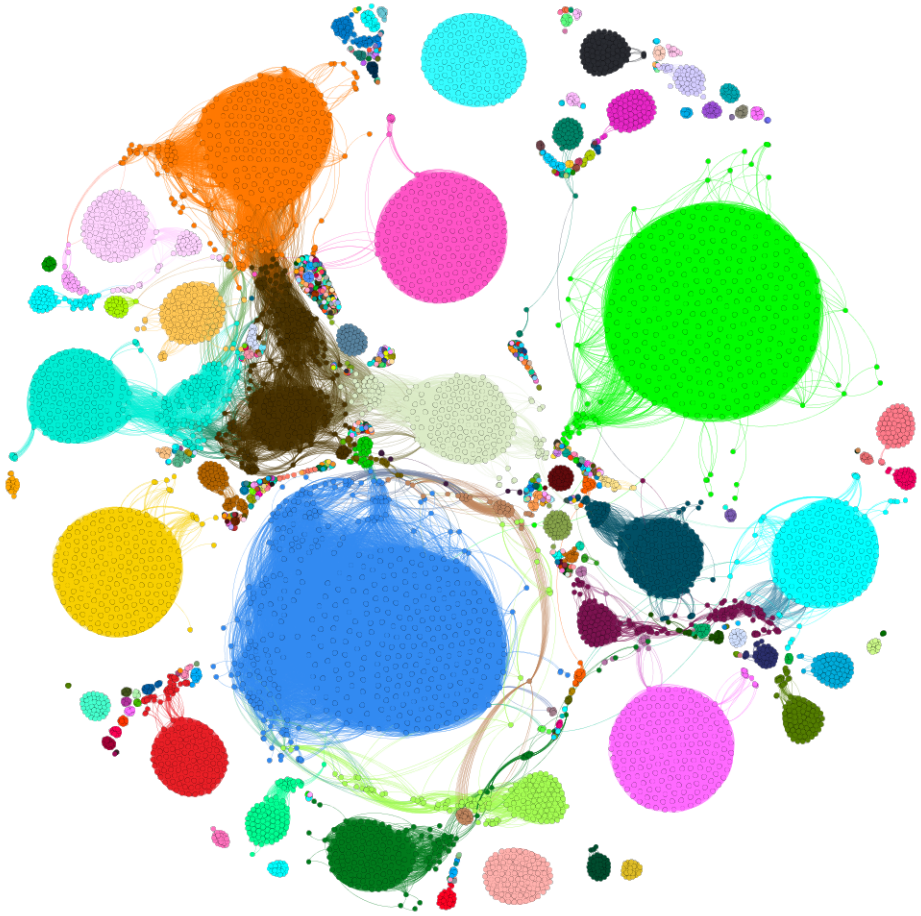


Figure 4.11: Cosine similarity of Entities and their winning procurements CPV codes

The large green cluster shown in figure 4.12 represents all companies that primarily won procurements that have specified the medical equipment CPV code.

Figure 4.12: Cluster medical equipment

One of the smaller clusters in red shown below in figure 4.13, represents the companies that have won procurements with furnishing and agricultural products CPV codes. These companies are mainly flower shops.

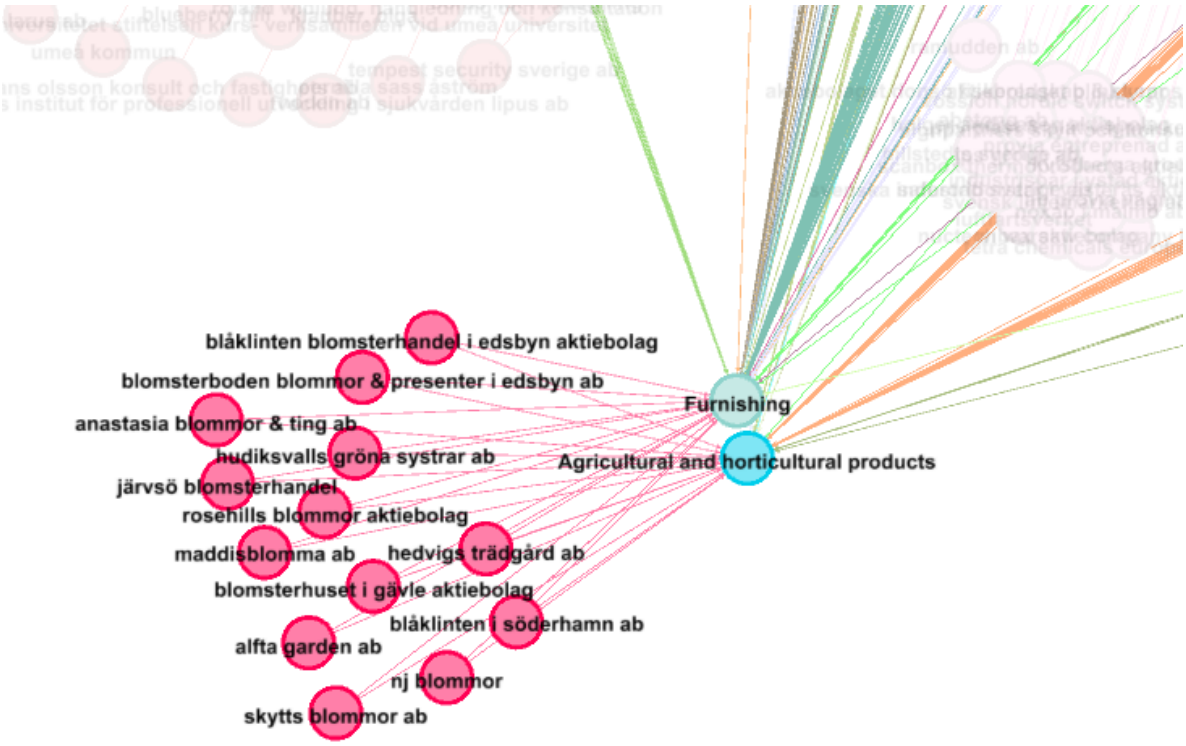


Figure 4.13: Cluster furnishing and Agricultural

One of the medium sized clusters (shown in figure 4.14) is the companies that have won procurements with Video services and Market economic research. It can be seen that media and advertisement companies are present which makes sense.

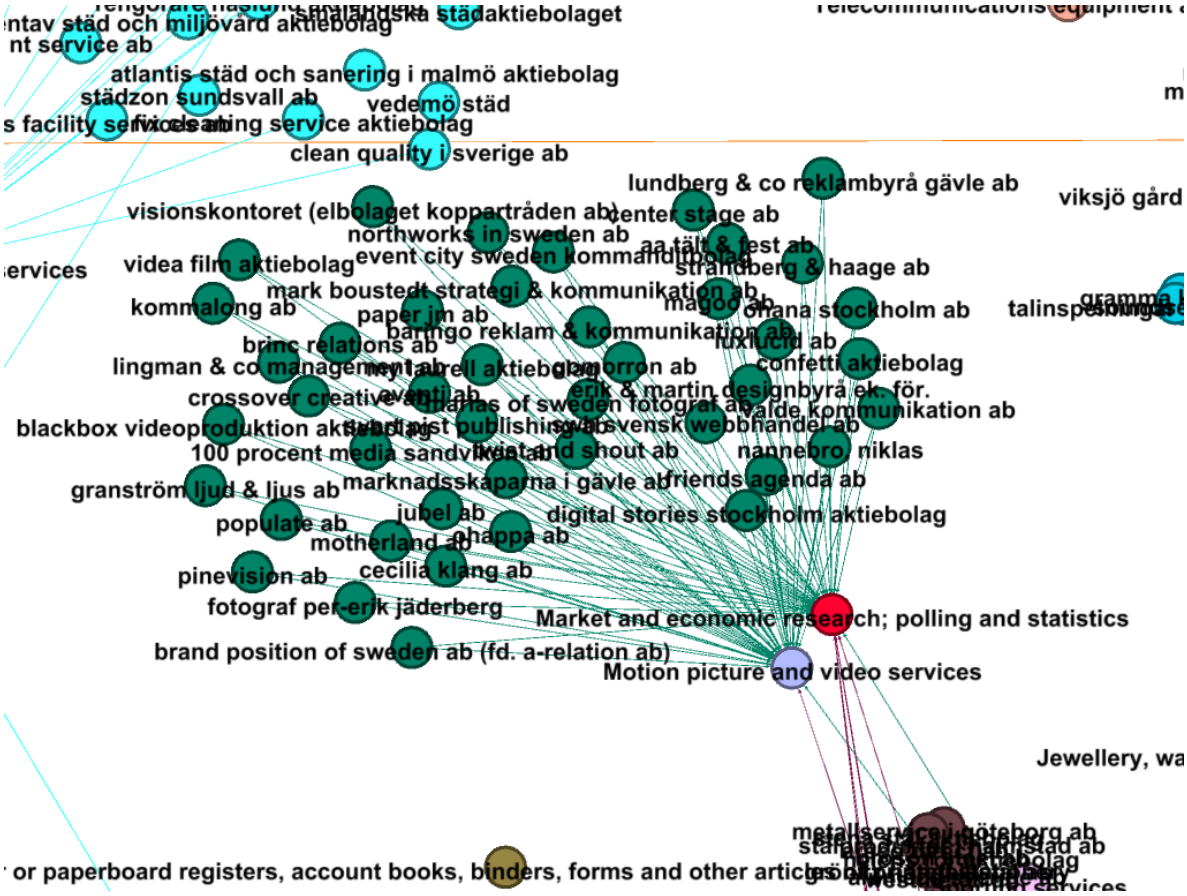


Figure 4.14: Cluster Market and economic research and Media

4.3 Centrality

4.3.1 Entity to Entity

The tables below are the results from applying the centrality algorithms mentioned in entity to entity in section 3.7.2. Table 4.1 below depicts the top scores with regard to degree centrality in descending order. Things to note are that the top entities are all procurers such as Göteborgs kommun, which has the highest degree followed by kommunalförbundet inköp Gävleborg and Trafikverket.

Entity	Degree centrality
Göteborgs kommun	388
Kommunalförbundet inköp Gävleborg	367
Trafikverket	361
Västra götalands läns landsting	321
Skåne läns landsting	301
Stockholms kommun	271
Uppsala läns landsting	265
Stockholms läns landsting	255
Västernorrlands läns landsting	222
Östergötlands läns landsting	220
Jönköpings läns landsting	219

Table 4.1: Degree of Entities to other Entities

Table 4.2 is the Closeness score in descending order in the same projection. It is clear that several companies have overtaken the procurers in centrality score compared to the degree centrality score such as WSP Sverige, Atea and Tyréns ab.

Entity	Closeness centrality
Wsp Sverige ab	0.401
Åf-infrastructure ab	0.377
Atea Sverige ab	0.372
Tyréns ab	0.364
Ramboll Sweden ab	0.356
Göteborgs kommun	0.351
Kommunalförbundet inköp Gävleborg	0.350
Norconsult ab	0.347
Trafikverket	0.347
Bravida Sverige ab	0.346

Table 4.2: Closeness centrality score of Entities to other Entities

In table 4.3 the Eigenvector centrality of the entities are presented. The scores are normalized by the maximum value. We can see that a majority of the highest scoring entities are procurers but there are still some companies that are competing for high centrality in the graph.

Entity	Eigenvector centrality (normalized by max score)
Västra götalands läns landsting	1.000
Skåne läns landsting	0.894
Wsp Sverige ab	0.809
Stockholms läns landsting	0.800
Göteborgs kommun	0.769
Uppsala läns landsting	0.751
Östergötlands läns landsting	0.714
Åf-infrastructure ab	0.685
Västernorrlands läns landsting	0.680
Jönköpings läns landsting	0.659

Table 4.3: Eigenvector centrality score of Entities to other Entities

4.3.2 Entity to NUTS County

Below in table 4.4 is the procurers' degree in the projection of procurer to county NUTS codes. Trafikverket has the highest degree meaning it has created procurements where at least 20 of them have different county NUTS codes.

Procurer	Degree centrality
Trafikverket	20
Kriminalvården	17
Migrationsverket	11
Skl kommentus ab	11
Fortifikationsverket	10
A-train aktiebolag	9
Sveriges domstolar	8
Försvarets materielverk	7
Specialfastigheter Sverige aktiebolag	7
Statens fastighetsverk	6

Table 4.4: Degree of Entity to Nuts

In table 4.5 the companies' degree in the projection of company to NUTS codes is shown. We can see the national reach companies have with this, these

companies have all been involved in procurements in over 15 different counties.

Company	Degree centrality
Åf-infrastructure ab	20
Samhall aktiebolag	19
Norconsult ab	19
Wsp Sverige ab	19
Ncc Sverige ab	19
Ramboll Sweden ab	18
Peab Sverige ab	17
Tyréns ab	17
Skanska Sverige ab	17
Ahlsell Sverige ab	17
Öhrlings pricewaterhousecoopers ab	16
Atea Sverige ab	16
Bravida Sverige ab	15

Table 4.5: Degree of Entity to Nuts

4.3.3 CPV to NUTS County

Presented in table 4.6 is the CPV codes' degree centrality in the projection of CPV Group to Nuts counties. Here we can see how different trades are national, e.g. Furnishing and Motor vehicles are more national in procurements than Miscellaneous repair and maintenance services.

CPV Group	Degree centrality
Business and management consultancy and relate...	21
Furnishing	21
Horticultural services	21
Motor vehicles	21
⋮	
Health services	20
Banking and investment services	20
Textile articles	19
Electricity, heating, solar and nuclear energy	19
⋮	
Adult and other education services	17
Support services for land, water and air transport	17
Miscellaneous repair and maintenance services	16
Sports goods and equipment	16

Table 4.6: Degree of Entity to Nuts

Chapter 5

Discussion

5.1 Data quality

As stated in the paper Assessing the potential for detecting collusion in Swedish public procurement presented in section 2.4.1, the TED data is lacking with regards to procurement values, bids and bidder names/identifier. This has resulted in procurements without winners if we could not identify or find a winner from the XML.

Since the data is not complete the results are more of an indication of what is possible to analyze with the TED data and only a rough evaluation of the Swedish procurement industry. For example, it is surprising that only some of the procurers have a degree of 10 in the entity to nuts shown in section 4.3.2. Which indicates a lack of data or there could also be a problem with the data that is existing. For example it can be unclear if the written Nuts codes are specific enough as it could be possible that an SE code might actually refer to a smaller set of county codes than all of the counties.

5.2 Graph model comparisons

The results of the model comparisons in 4.1 shows that an appropriate model for procurements and tenders is to have the data stored in more nodes and relationships as this drastically increases the query speed and creates more possibilities to create projections that can be used for analysis. All though the

downside is an increase in storage size, it is generally a worthy trade off as it is often easier to increase the storage available than increasing speed. What is surprising is that the more expansive model is as fast as when indexing the first model. It makes sense that using relationships over properties is quicker as we are storing the data in a database that is optimized for node to node traversal. If a property can be shared among several nodes it can be advantageous to have the property in a separate node. Whereas information that is specific to a node should be stored as a property. This makes every relationship in the database relevant as we can traverse over many different relationships to find what we are looking for, it also allows for more specific queries.

5.3 Graph analysis

With the database setup, the graph modeled and the data migrated, it was possible to find the largest influencers and clusters of nodes such as companies and procurers. In section 4.2 we identified clusters of nodes in several different projections. Such as in the entity to entity projection where we showed clusters such as the Landstings and the Universities. These clusters are entities that work with similar companies/procurers. We could also identify the most central nodes such as Göteborgs kommun and WSP Sverige which shows that they have great influence within the entity to entity graph.

It was also possible to extrapolate more specific patterns by using a projection to create new relationships based on similarities of the projection. With those newly created relationships we could retrieve a new projection to identify clusters of nodes and central nodes. Such as the similarity-based clustering in section 4.2.3.

The Louvain algorithm has primarily been used as it worked well with the larger graphs and generated well-formed clusters based on intuitive observations and modularity scores. A Markov clustering implementation was tried but the implementations showed to be too computationally heavy for larger graphs. Similarly, the computation time when trying to calculate the Betweenness centrality proved to be too long. One issue during this thesis has been on how to interpret the results of the clustering and how to evaluate the clusters. Outside of visualizing the graph and getting an intuitive idea of the clusters, modularity score was used as a guidance to examine the correctness of the clusters.

Overall the graph analysis shows great promise of what patterns that are possible to be found with both basic and more elaborate models. E.g. a simple projection such as the Entity to Entity in 3.7.2 gives insight into the structure of the area of procurement. By breaking out some of the properties as separate nodes it is possible to get even more detailed information from the procurement data as shown in SNI to CPV. With extra steps it is also possible to easily evaluate node similarity and use that similarity to create clusters of the nodes.

5.4 Example Use Cases

The Centrality of the entities is used to evaluate how connected and influential a company is in the sub graph of entities. If a company is highly connected, we know that it is taking money from a lot of pots, which other companies want to emulate to expand their business. With the combination of a similarity graph and the entity to entity graph you could find companies that are similar in the category of their winnings, but one company is able to win from several different suppliers whereas another is only winning from one because they lack information about possible procurements. The lesser company could then look into the more connected companies to find information about which procurers they often win procurements from.

Being able to visualize the network of procurements can be eye opening as one can build an intuitive view of how companies and procurers are connected. For example. which communities exist such as the universities and which companies work towards the universities.

Another use case is if we are investigating a specific area of public procurements, the first agencies to visit are the ones that are more influential in that area. If they are very connected or influential, they are more likely to have more extensive and relevant information.

Creating a translation between SNI to CPV is challenging when only looking at their descriptions. But with the ability to project the sub-graph of SNI codes to CPV codes it is possible to find which SNI and CPV codes that often occur together and get an indication of which SNI is associated with which CPV.

5.5 Threats to validity

There are a few possible threats to validity.

The first one being the verification of migration of the data. As it was verified manually by counts of records and property checks on a smaller subset of the parsed data.

Secondly, when running the queries in section 3.6.1 the database makes use of caching which imitates the database in a production setting. If we examine the speed without caching and focus on the IO and disc read speeds, we would have to set the cache size to zero or disable it.

Thirdly, even though modularity is a measure of how well a graph is clustered, it is not regarded as definitive which has been shown in the paper Bad communities with high modularity [44]. They reached the conclusion that modularity is not always the best indicator that the clustering is optimal. The resolution limit of Louvain clustering means that it can have trouble detecting small clusters in large networks.

Lastly, as mention in section 5.1, the data is incomplete and can be inaccurate for some procurements.

5.6 Sustainability

Exploring the usage of graph databases and procurements gives a lot of transparency in the procurement market. It becomes easier to find relationships between different companies, procurers and procurements. This ease of use of finding patterns and relationships saves time and money for both procurers and companies that are looking to investigate the procurement market. This thesis might also provoke interest in others to see what the procurement market looks like today and how much money it involves. Since public procurements involve such huge amounts of money it is important to be able to understand the market. If a fraction of the money spent on procurements could be saved by procuring agencies, that is money that could be spent on other areas such as health care and education.

Chapter 6

Conclusions

This thesis is a proof of concept of what can be done with procurements in graph databases. The results showed that modeling the data in more nodes provides greater speed at the disadvantage of larger storage size. Compared to just indexing, more nodes and edges also provides the possibility to apply centrality and community detection algorithms.

By storing data in graph structure, it is possible with the usage of projections to retrieve specific sub-graphs from the database in a convenient way, such as the Procurer to Company and SNI code to CPV code. With the ability to get these projections it is possible to identify specific patterns in the data with the use of graph analysis, specifically centrality and community detection in this case. We can determine which procurer or companies that are more influential and more geographically spread. Additionally, we also have the capability to find different communities of procurers and companies based on how similar they are in trade and their transactions between each other.

One of the obstacles is the issue of the public procurement and tender data being lacking, which has been raised both in the background and discussion sections. With a more complete data set it would give a more accurate representation of the public procurement market. It should also be possible to extrapolate even more interesting clusters and centralizes such as relationships between edges with a weight based on the procurement's value and bids.

In conclusion a graph database structure shows great potential for finding patterns in the area of procurements.

6.1 Future work

There is still much to be explored in the combination of procurements and graph databases. One improvement could be to get more qualitative and quantitative data. The data used in this thesis could be further improved with more detailed information such as the economic values of bids, what bids were made and by which company. Finding more data sources would be one way of achieving this.

This thesis has explored a couple of ways of modeling procurements and tenders, there are still a lot of different models worth considering. For example, a person centred models where the focus is more on individuals and their involvement in procurements. Or perhaps a more granular representation of the whole procurement and tender process is suitable. We can create a timeline of sorts for each procurement, by splitting up procurements into the different phases of the procedure where we look at documents involving the announcement, bids, evaluation and awards.

There is still a lot of undiscovered uses of graph analysis that can be done with the current models. This thesis has only touched the tip of the iceberg of what is possible. With the ability to traverse between nodes, create connections between them that are not explicitly set and then create a projection into a new graph we can find an endless number of interesting graphs to analyze.

Finally, it would be very interesting to create a graph database that can be used in production and continuously be filled with all the newly created procurement data and data involved around them. This would help people involved in the procurement industry, allowing for queries that focuses on the relationships.

Bibliography

[1] Codd Edgar. “A relational model of data for large shared data banks”. eng. In: *Communications of the ACM* 26.1 (1983), pp. 64–69. ISSN: 1557-7317.

[2] R. Kumar Kaliyar. “Graph databases: A survey”. In: *International Conference on Computing, Communication Automation*. May 2015, pp. 785–790. DOI: 10.1109/CCAA.2015.7148480.

[3] European Commission. *Public Procurement*. 2019. URL: https://ec.europa.eu/growth/single-market/public-procurement_en.

[4] Andreas Larsson (ansvarig handläggare) and Annika Töyrä. “Statistik om offentlig upphandling 2018”. swe. In: (2018).

[5] Statistics Sweden. *Sökning efter SNI-kod*. 2019. URL: http://www.sni2007.scb.se/default.asp.

[6] Publications Office of the European Union. *CPV*. 2019. URL: https://simap.ted.europa.eu/web/simap/cpv.

[7] *Commerce*. URL: https://www.visma.se/commerce/.

[8] Publications Office of the European Union. *European public procurement*. 2019. URL: https://simap.ted.europa.eu/european-public-procurement.

[9] David. Dominguez-Sal et al. “Survey of graph database performance on the HPC scalable graph analysis benchmark”. In: vol. 6185. 2010, pp. 37–48. ISBN: 3642167195.

[10] 2019 May 23, 2019 May 20, and 2019 May 17. *Neo4j Graph Platform – The Leader in Graph Databases*. URL: https://neo4j.com/.

[11] *Apache Jena* -. URL: https://jena.apache.org/.

[12] *A Graph Database*. URL: http://www.hypergraphdb.org/.

[13] *High-performance human solutions for Extreme Data*. URL: http://www.sparsity-technologies.com/.

[14] Salim Jouili and Valentin Vansteenberghe. "An Empirical Comparison of Graph Databases". eng. In: *2013 International Conference on Social Computing*. IEEE, 2013, pp. 708–715. ISBN: 9780769551371.

[15] *TITAN*. URL: https://titan.thinkaurelius.com/.

[16] *Graph Database | Multi-Model Database*. URL: https://orientdb.com/.

[17] solid IT. *db-engines*. 2019. URL: https://db-engines.com/en/ranking/graph+dbms.

[18] Ian Robinson, Jim Webber, and Emil Eifrem. *Graph Databases 2nd Edition*. O'Reilly, 2015. ISBN: 978-1-491-93089-2.

[19] Rik Van Bruggen. *Learning Neo4j*. eng. Olton: Packt Publishing, Limited, 2014. ISBN: 9781849517164.

[20] Linton C. Freeman. "Centrality in social networks conceptual clarification". eng. In: *Social Networks* 1.3 (1978), pp. 215–239. ISSN: 0378-8733.

[21] Mark. Needham and Amy. E. Hodler. *Graph Algorithms: Practical Examples in Apache Spark and Neo4j*. O'Reilly Media, Incorporated, 2019. ISBN: 9781492047681. URL: https://books.google.se/books?id=UwIevgEACAAJ.

[22] Linton Freeman. "A Set of Measures of Centrality Based on Betweenness". eng. In: *Sociometry* 40.1 (1977). ISSN: 0038-0431. URL: http://search.proquest.com/docview/1297089201/.

[23] L. Page et al. "The PageRank citation ranking: Bringing order to the Web". In: *Proceedings of the 7th International World Wide Web Conference*. Brisbane, Australia, 1998, pp. 161–172. URL: citeseer.nj.nec.com/page98pagerank.html.

[24] R. V. Mises and H. Pollaczek-Geiringer. "Praktische Verfahren der Gleichungsauflösung ." In: *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik* 9.2 (1929), pp. 152–164. DOI: 10.1002/zamm.19290090206. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/zamm.19290090206. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/zamm.19290090206.

[25] Robert Tarjan. “Depth-First Search and Linear Graph Algorithms”. eng. In: *SIAM Journal on Computing* 1.2 (1972). ISSN: 00975397. URL: http://search.proquest.com/docview/920020233/.

[26] Stijn. Dongen. “Graph Clustering by Flow Simulation”. eng. In: (2000).

[27] Venu Satuluri and Srinivasan Parthasarathy. “Scalable Graph Clustering Using Stochastic Flows: Applications to Community Discovery”. In: *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’09. Paris, France: ACM, 2009, pp. 737–746. ISBN: 978-1-60558-495-9. DOI: 10.1145/1557019.1557101. URL: http://doi.acm.org/10.1145/1557019.1557101.

[28] Deepayan Chakrabarti and Christos Faloutsos. “Graph Mining: Laws, Generators, and Algorithms”. In: *ACM Comput. Surv.* 38.1 (June 2006). ISSN: 0360-0300. DOI: 10.1145/1132952.1132954. URL: http://doi.acm.org/10.1145/1132952.1132954.

[29] M. Newman. “Equivalence between modularity optimization and maximum likelihood methods for community detection”. In: *Physical Review E* 94 (Nov. 2016). DOI: 10.1103/PhysRevE.94.052315.

[30] Vincent D Blondel et al. “Fast unfolding of communities in large networks”. eng. In: *Journal of Statistical Mechanics: Theory and Experiment* 2008.10 (2008). ISSN: 1742-5468.

[31] Andrea Lancichinetti and Santo Fortunato. “Community detection algorithms: A comparative analysis”. In: *Phys. Rev. E* 80 (5 Nov. 2009), p. 056117. DOI: 10.1103/PhysRevE.80.056117. URL: https://link.aps.org/doi/10.1103/PhysRevE.80.056117.

[32] Zhao Yang, René Algesheimer, and Claudio J. Tessone. “A Comparative Analysis of Community Detection Algorithms on Artificial Networks”. In: *Scientific Reports* 6.1 (2016). ISSN: 2045-2322.

[33] Mihály Fazekas and Bence Tóth. “Assessing the potential for detecting collusion in Swedish public procurement”. swe. In: (2016). ISSN: 1652-8069.

[34] Thashen Padayachy, Brenda Scholtz, and Janet Wesson. “An Information Extraction Model Using a Graph Database to Recommend the Most Applied Case”. eng. In: *2018 International Conference on Computing, Electronics ‘I&’ Communications Engineering (iCCECE)*. IEEE, 2018, pp. 89–94. ISBN: 9781538649046.

[35] George Stergiopoulos et al. “Risk mitigation strategies for critical infrastructures based on graph centrality analysis”. eng. In: *International Journal of Critical Infrastructure Protection* 10 (2015), pp. 34–44. ISSN: 1874-5482.

[36] Amazon. *Amazon EC2 Secure and resizable compute capacity in the cloud. Launch applications when needed without upfront commitments.* 2019. URL: https://aws.amazon.com/ec2/.

[37] Amazon Web Services. *Amazon EBS Volume Types*. 2019. URL: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSVolumeTypes.html.

[38] Michael Hunger. *APOC User Guide 3.5*. 2019. URL: https://neo4j-contrib.github.io/neo4j-apoc-procedures/.

[39] Michael Hunger. *Awesome Procedures On Cypher – APOC*. 2019. URL: https://neo4j.com/labs/apoc/.

[40] Mark Needham ‘I&’ Amy E. Hodler. *The Neo4j Graph Algorithms User Guide v3.5*. 2019. URL: https://neo4j.com/docs/graph-algorithms/current/.

[41] Gephi. *Gephi makes graphs handy*. 2019. URL: https://gephi.org.

[42] Douglas Comer. “Ubiquitous B-tree”. In: *ACM Computing Surveys (CSUR)* 11.2 (1979), pp. 121–137.

[43] Neo4j. *Neo4j Bolt Driver 1.7 for Python*. 2019. URL: https://neo4j.com/docs/api/python-driver/current/.

[44] Ath Kehagias and Leonidas Pitsoulis. “Bad communities with high modularity”. In: *The European Physical Journal B* 86.7 (2013), p. 330.

Appendix A

Ted XML structure

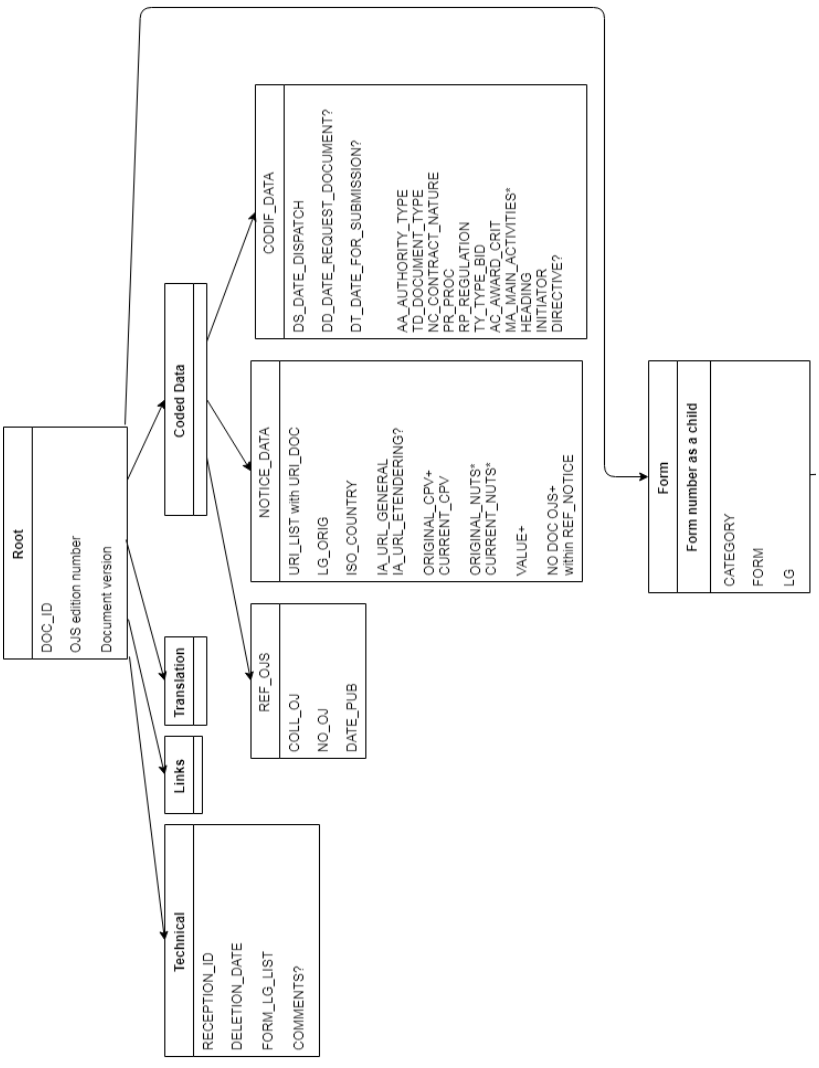


Figure A.1: Ted XML structure part 1.

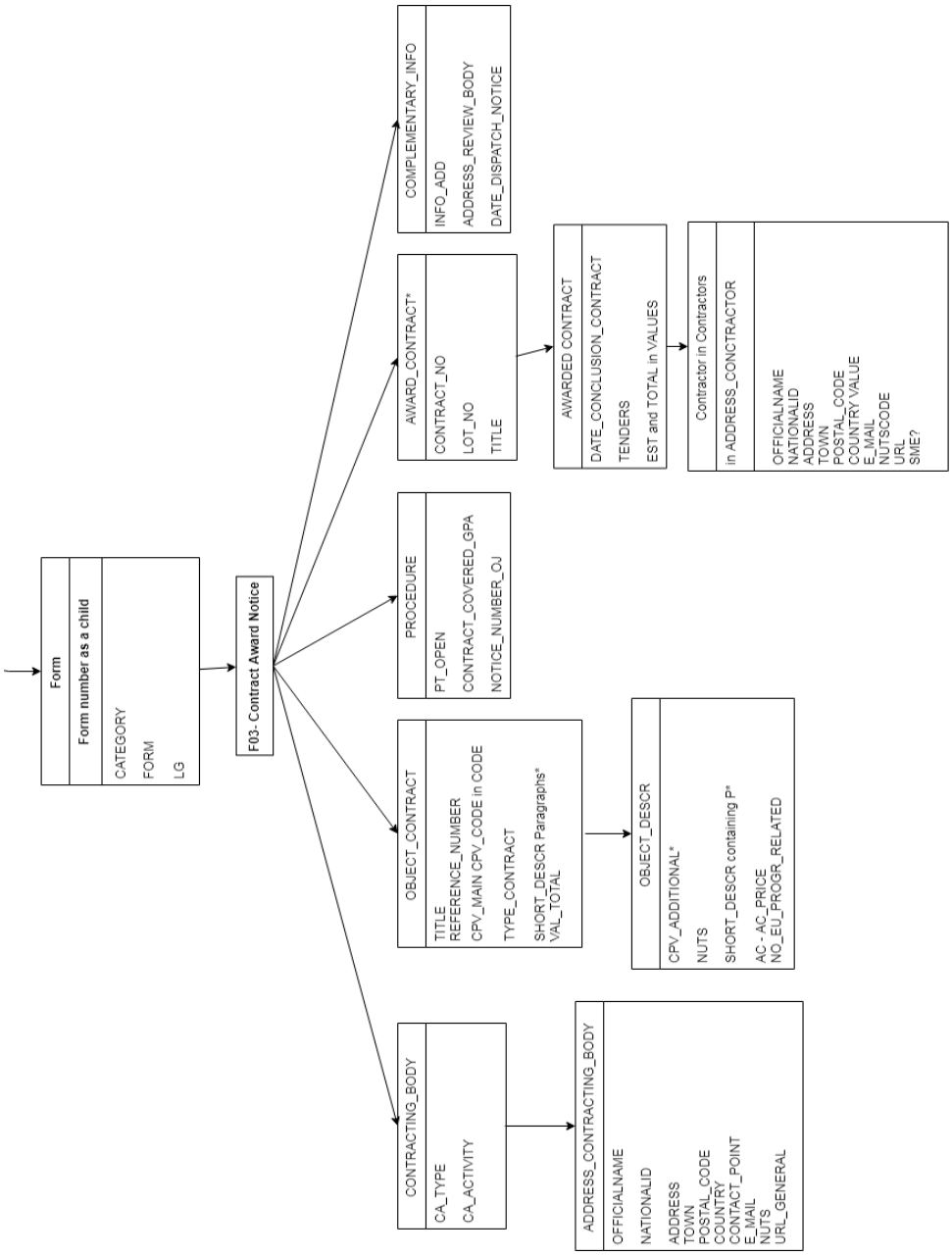


Figure A.2: Ted XML structure part 2.

TRITA-EECS-EX-2019:833