**Analysis of Large Scale Social Networks**

Graph Database Project:

Neo4J graph Database Metadata Management Prototype

François Wilberz R0771918

Nicolas Delahousse R0789028

Robbe Neyns R0674734

Contents

[1 Context 3](#_Toc41259886)

[2 Objectives 4](#_Toc41259887)

[3 Graph database model 4](#_Toc41259888)

[Model structure - Nodes and properties 4](#_Toc41259889)

[Model structure - Relationships 5](#_Toc41259890)

[4 Analysis 6](#_Toc41259891)

[4.1 Monitoring communities and users 6](#_Toc41259892)

[Community monitoring 7](#_Toc41259893)

[Collaboration between communities 7](#_Toc41259894)

[Additional ad hoc analysis 8](#_Toc41259895)

[4.2 Impact Analysis 9](#_Toc41259896)

[4.3 Centrality 10](#_Toc41259897)

[5 Results 11](#_Toc41259898)

[6 Conclusion with focus on gained knowledge 12](#_Toc41259899)

[7 Individual contribution: 13](#_Toc41259900)

[7.1 François Wilberz 13](#_Toc41259901)

[7.2 Nicolas Delahousse 14](#_Toc41259902)

[7.3 Robbe Neyns 14](#_Toc41259903)

[8 Appendix 14](#_Toc41259904)

# 1 Context

With the increasing availability of sizable amounts of data and large advances in computer performance over the last few decades, research in artificial intelligence and big data has been able to make spectacular progress. The progress made by scientists has led to many new applications that could revolutionize our society. Self-driving cars, service automation and face recognition are just a few examples among so many others. However, the performance of many of those applications often relies on the amount of data collected as well as on their quality. The increasing amount of data is also in need of regulation, personal information is at the core of many concerns regarding privacy and human rights as they might be used at the expense of the individual. To counter possible exploitation by corporations or smaller entities, several regulations have emerged to ensure that the privacy and rights of individuals are properly regulated and protected. Among those, the General Data Protection Regulation (GDPR) from the European Union which is still today considered as a reference in the field of data protection. However, these much needed regulations tend to complexify the efficient use of data within an organization even more. .

As a result, today more than ever, it is important to have strong data governance in place in organizations to ensure proper use of personal information to avoid legislative problems, and to improve the efficiency of organizations. Indeed, a good data strategy should provide an organization with the appropriate resources to:

- Have a common understanding of their data

- Break the data silos from the organization (work as one entity and not by department)

- Improve data quality

- Assign ownership and responsibilities to their data asset

Among the existing tools, we will focus on metadata management software. Such tools collect and organize the metadata and interdependencies of an organization’s data assets in a centralized repository in the form of a graph database. While information describing the asset is important, the interaction between them also represents an important source of information for the organization. For example, a query uses tables to create views or the definition of a term impacts the columns storing the information defined. In this project, we will attempt to partially reproduce such a repository by using as a starting point the AdventureWorks2017 database.

# 

# 2 Objectives

Given the scope of the assignment, we only cover some of the numerous functionalities provided by a metadata management tool. To create these functionalities, we rely on graph database technology:

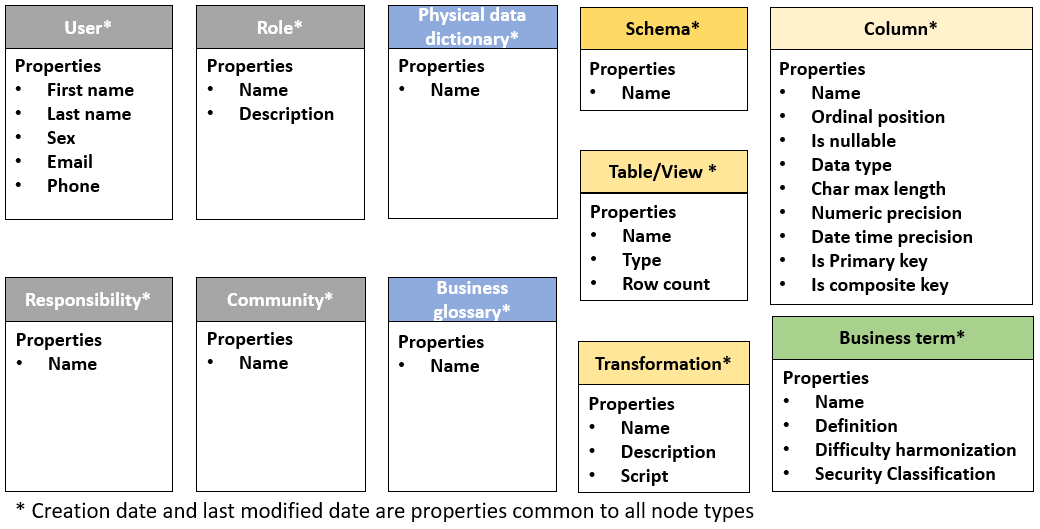
* **Monitoring communities and users:** Organizations usually do not act as one centralized entity; Different communities and departments exist, each with their own field of expertise. It is therefore needed that ownership and responsibilities over data are split between departments; For instance, HR maintains control over the HR data. However, resistance to change from employees, or fear of sharing information with other departments are not uncommon when trying to implement such software in an organization. Through the use of a graph database, it would therefore be possible to generate a view on the number of data assets owned by a community, and the number of assets created recently (by using property creation date) etc. In other words, a graph database of the metadata would facilitate the monitoring of good data practices within the organization.
* **Impact analysis**: An organization is not a static environment; it is condemned to evolve. However, it might happen that a change request, for example regarding a definition, considerably impacts the organization due to the underlying changes on other assets depending on that definition. Some implications might be unknown when the decision is made. In the end, these implications could lead to more harm than good as they might require that numerous changes be made in various systems. We will therefore showcase one example in this report on how graph databases can help solve this issue and can support decision making in a management or IT context.
* **Centrality of data assets**: The more a node is linked to other nodes, the more likely it contains important metadata for an organization. If badly defined, the impact on the organization will be more important. To showcase this, we will use different centrality algorithms from Neo4j and explain their respective use.

# 3 Graph database model

In this section, we explain how we approached the design of our model, with the objective of capturing the metadata from the data contained in the AdventureWorks database. The AdventureWorks database is a Microsoft product that is based on a fictitious manufacturing company of the same name.

### Model structure - Nodes and properties

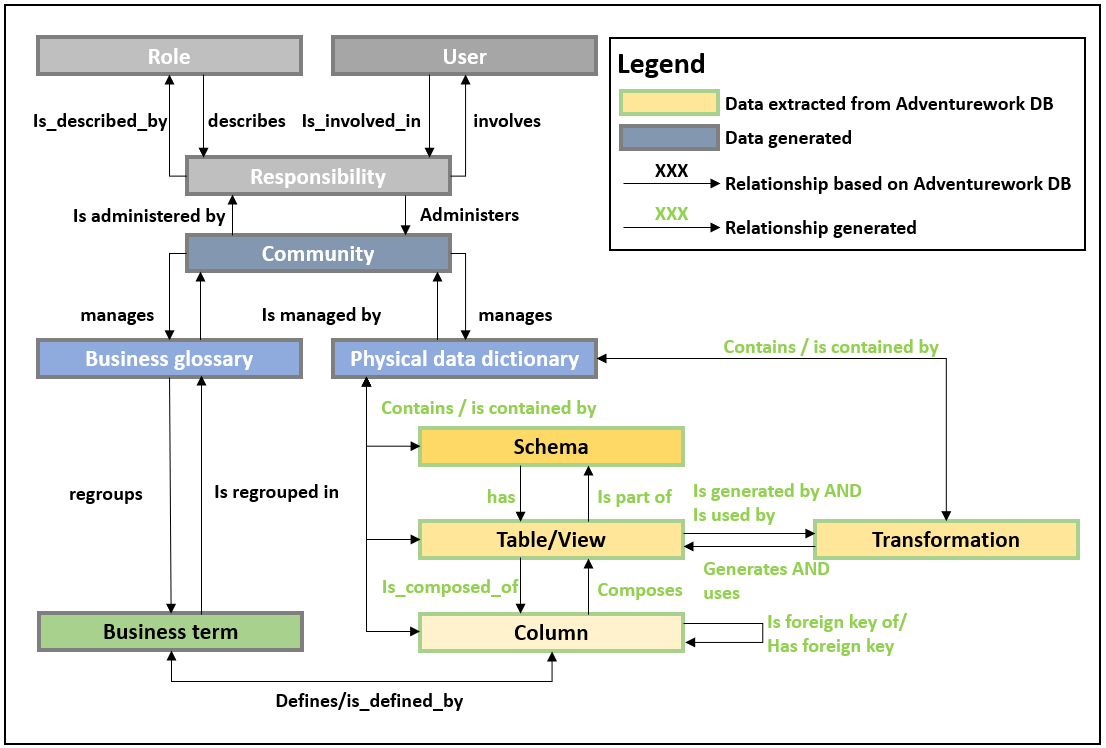
Inspired from the approach of the Collibra asset model, we implemented a smaller model with 11 node types and 24 relationship types. Each asset type is listed in Figure 1 with its properties and was selected based on the objectives we wanted to achieve through this work. All yellow node types data were extracted from the AdventureWorks database. Queries used to do this can be found in appendix A. The rest of the data were generated by ourselves while taking into account the AdventureWorks metadata. All transformed data files were saved as CSV files. provides a good overview of the data generated by ourselves and the data extracted from AdventuresWorks database.



**Figure 1** Asset Types

### Model structure - Relationships

A metadata management tool should be accessible to everyone within the organization, such that the impact of a type of node on another should easily be understood by everyone. Therefore, each relation between two nodes is represented by 2 directed edges, explaining the influence of the first node on the second one and vice versa. The final meta graph is displayed in Figure 2 .



**Figure 2** Final Meta Graph

# 4 Analysis

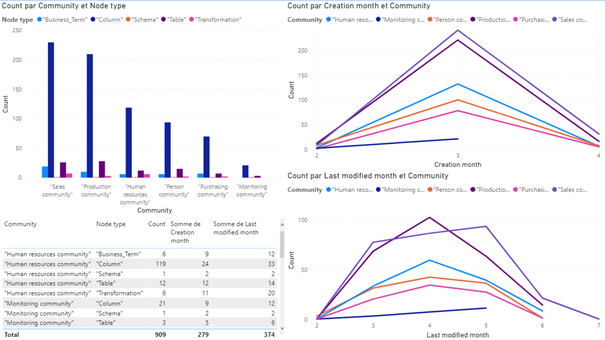
## 4.1 Monitoring communities and users

A metadata management tool is a repository in which ideally all information about the data of an organization should be centralized in a structured way. Within the tool, the metadata should be accessible to everyone unless specified otherwise for security reasons. Nevertheless, it is still important that the content, the quality and the maintenance of each data asset (column, table, business term and transformation) is guaranteed by the relevant - and appropriate - people within the organization. Organized in communities, each group member has different roles based on their responsibility in the community. For example, the owner will be held accountable for the data quality and the proper governance of his assets while the technical steward will be in charge to update, maintain and document the systems of his community based on the requests of the business.

Those communities are generally defined based on existing groups within the organization such as department or geographic zones. From personal experience, the adoption of such new technology is not an easy process, many mistakes can occur by a lack of understanding and commitment. This is why monitoring the progress of each community and their interaction with the other communities is a great way to ensure a proper implementation of such a technology. We will show in the next three subsections different approaches in how graph databases could be leveraged in our use case.

### Community monitoring

Via the query in appendix B we have been able to extract data containing the evolution of the creation and modification of data assets per community and per data type. While this data is not easily comparable in Neo4J we have used the Power BI custom connector to directly collect the aggregated data in the BI platform and build a dashboard that could help management to track the evolution of each community (Figure 3, Appendix X).



**Figure 3** Community Monitoring Dashboard

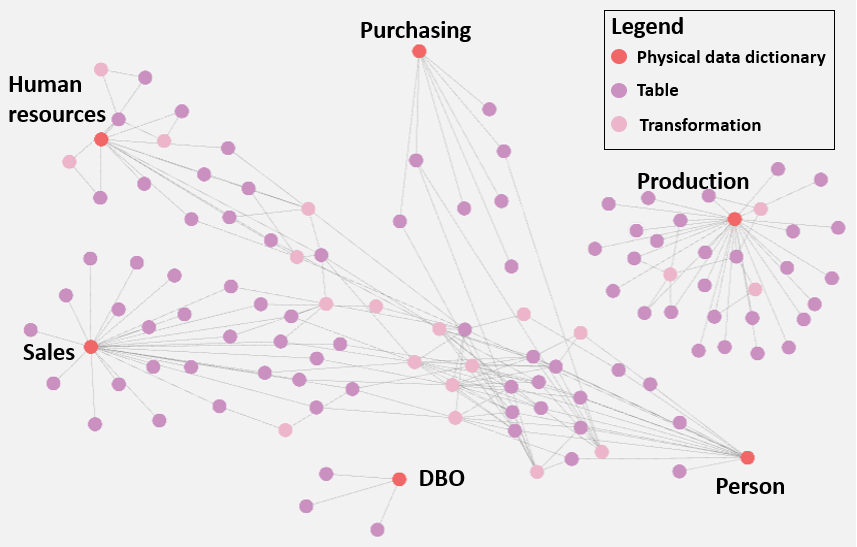
We highlight the following takeaways:

* Most of the assets were created in March.
* The Sales and the Production community are the one owning the most assets.

However, besides these general observations, the take-away of this exercise is really the possibility to, and added value of, integrating the Graph database with other tools, such as Power Bi.

### Collaboration between communities

Breaking data silos within an organization is a must, when an organization wants to leverage their data to its full potential. While there exist several approaches to visualize the collaboration between communities, the creation of views using tables from different communities could be used as an collaboration indicator. To do so, we used Neo4J Bloom; Our results can be found in figure XXX. It can be clearly observed that tables managed by the production community are not used to create trans department views. Such views could provide insights that could benefit the collaboration between departments such as a view showing the evolution of the inventory based on the sales or purchase orders. This could be used for predictive analysis or planning.



**Figure 4** Visualization of the collaboration between communities to create views

Such an approach can be used to find new or missing synergies within an organization but not to draw conclusions. Indeed, it might happen that collaboration already exists but is not properly registered in the system for example.

### Additional ad hoc analysis

While communication between different communities is considered essential for a smooth process, the ownership and responsibilities on data assets should in general be in charge of the community responsible and people should generally only have roles in their community of origin. The query in figure 5 visualizes all employees that possess roles in more than one community.



**Figure 5** Responsibilities of system users across communities

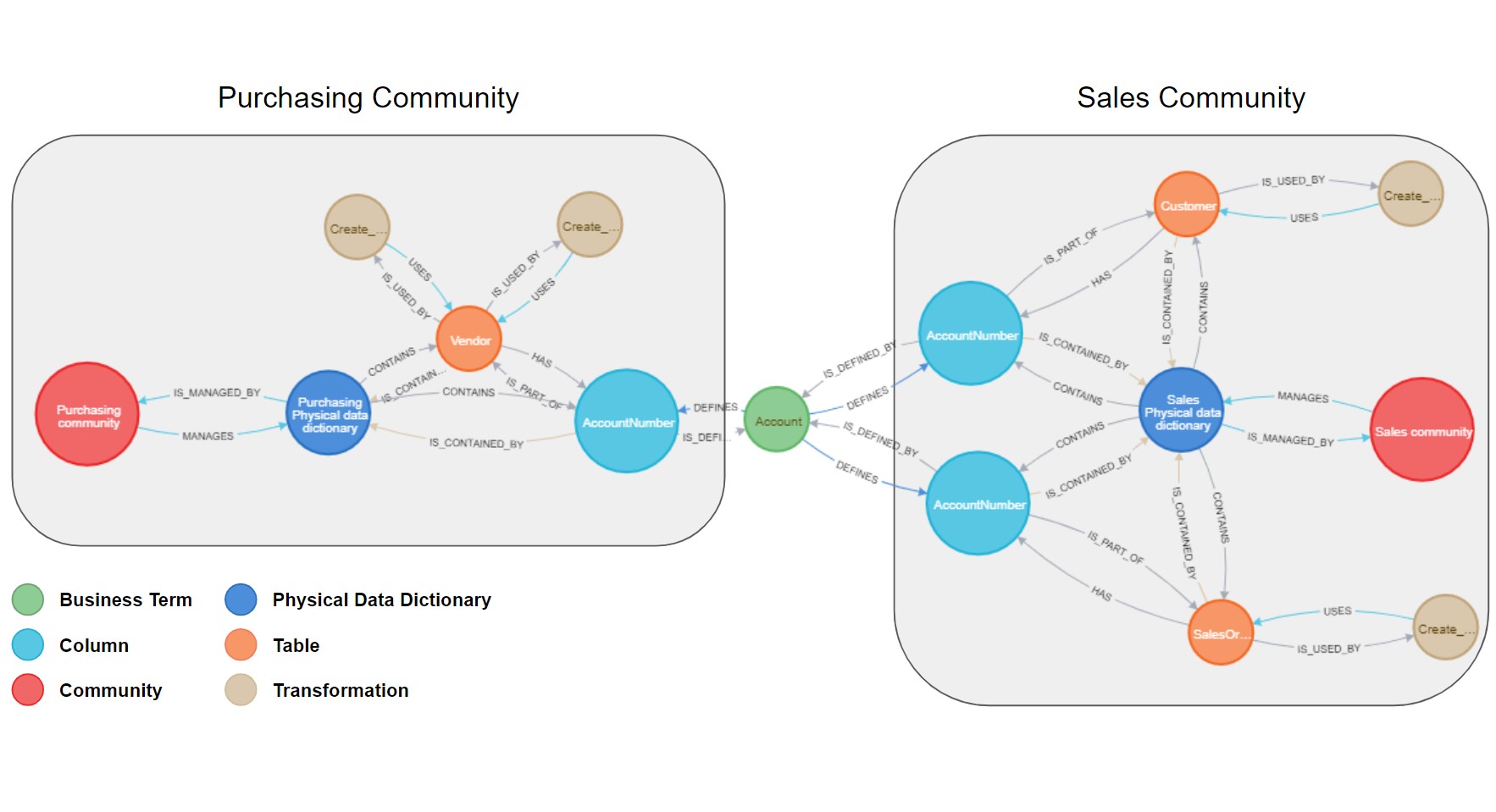
## 4.2 Impact Analysis

In this section we demonstrate the clear benefits of social networks in performing impact analysis. The following is a typical scenario where a ‘simple’ technical request is made by non-technical employees.

**Scenario**

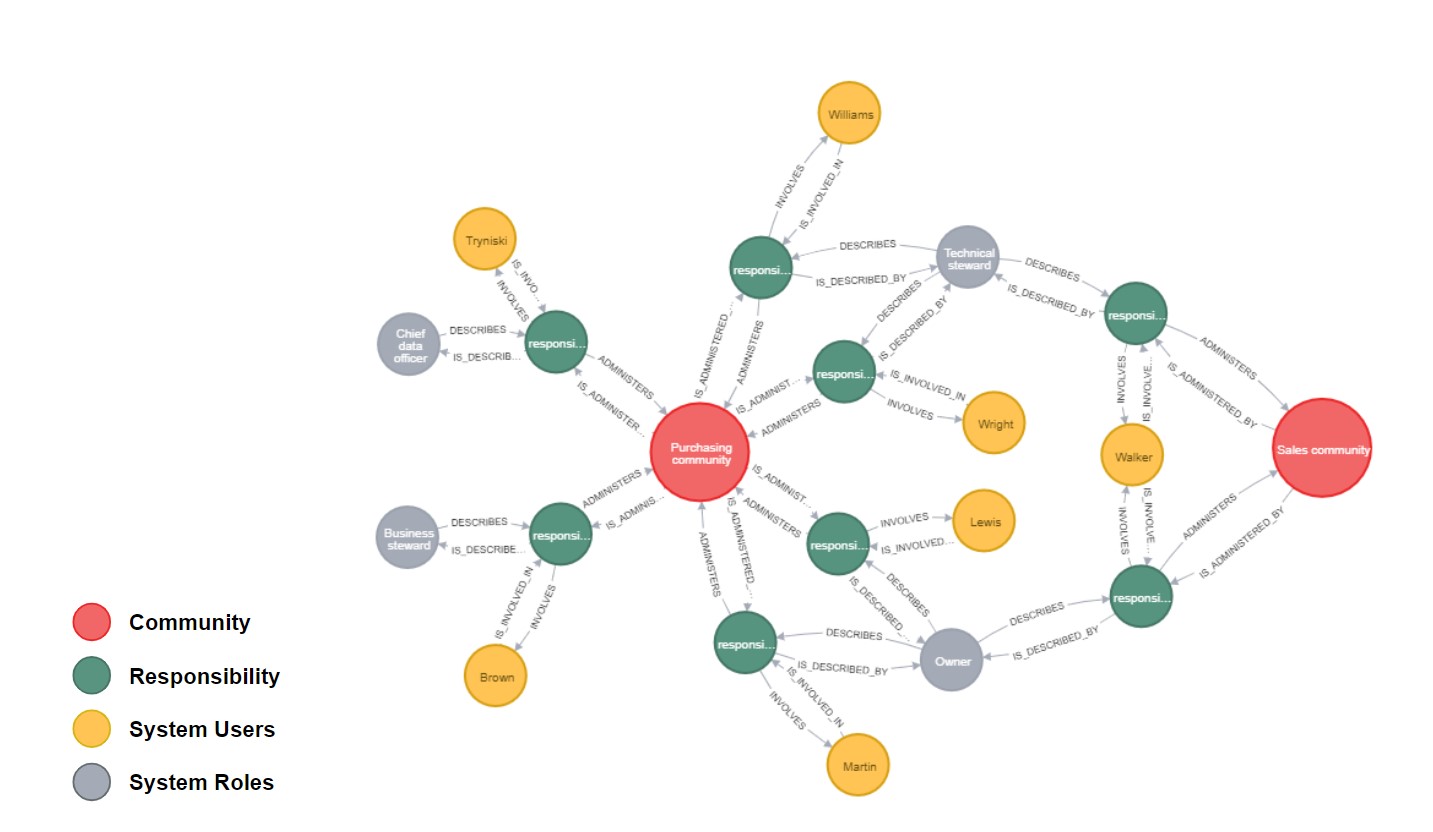
Sales people request to change the definition of the business term `Account` in order to better reflect the reality of the company. Before proceeding, they wish to know if a change in the definition of “Account” would impact components outside the Sales community.

As is shown in figure 6, the Account defines an AccountNumber column for the Customer, SalesOrderHeader and Vendor tables respectively. The Customer and SalesOrderHeader tables belong to the Sales community, while the Vendor table belongs to the Purchasing community. It is clear that further analysis is required to determine what the impact of a change in Account definition will be on the Purchasing community.



**Figure 6** Link between the purchasing and the sales community

This analysis will require grouping the relevant people from both departments. In our scenario, the relevant authorities on such matters are called technical stewards. The technical stewards of both departments can identify each other (Appendix F) and organize a meeting as is shown below in figure 7.



**Figure 7** Users that share responsibilities across communities

The advantages of the graph database are clear in this context. A company that does not use a graph database to visualize its data lineage could not perform this analysis as effectively. While being a simple scenario, it effectively reflects the reality of many companies that are isolated in the departments and have to make individual - but impactful - decisions.

## 4.3 Centrality

**Theory**

Centrality algorithms are used to determine the importance of distinct nodes in a network. Different centrality analysis algorithms exist in the Neo4J graph database environment, an overview of them can be seen inTable 1. Three algorithms were applied to the Adventure works graph database: the degree centrality, the ArticleRank and the closeness centrality . Each measure reveals information about a different aspect of the graph, and the relative importance of specific nodes. The degree centrality provides insights concerning which nodes have the most connections, it is thus a measure of local influence. The pageRank centrality on the other hand gives a broader idea concerning the connectivity of a node, as it also takes the direction of the connections into account. Finally, the closeness centrality informs us which nodes are closest to all other nodes in the graph.

The different centrality measures are used to investigate the importance of the various business terms in the graph. This is a realistic use case since it gives an insight into which business-terms – and their respective definitions - have most influence on the proper functioning of the organisation. In other words, if a term or its definition is changed we have an immediate idea of the influence of this change on the graph and to which terms we should pay most attention.

Another type of node to which these metrics are applied is Columns. This is useful for several reasons, for example if the organization is expanding, multiple columns will be recurrent and a naming convention is required. Executing a centrality analysis shows which columns are most important and hence need to be prioritized when it comes to naming and data-type rules.

Finally, the closeness centrality is applied to the node type ‘sys-users’, to get an idea which users are able to spread information efficiently through the graph. For system users, the other two measures did not provide very insightful results since they only have one direct connection (to their responsibility).

|  |  |
| --- | --- |
| *Centrality measure* | *How it works* |
| **Page Rank** | Assigns importance based on the number of links held by each node and on how well connected a node and its neighbours are. Takes also weight and direction of links into account |
| **ArticleRank** | Variant of Page Rank. However, the assumption that nodes with a low out-degree are more important is weakened in ArticleRank |
| **Betweenness Centrality** | Measures the number of times the node lies on the shortest path between two nodes. The shortest path is calculated between every pair of nodes using the breath first algorithm |
| **Closeness Centrality** | Assigns importance based on the ‘closeness’ to other nodes in the network |
| **Degree Centrality** | Assigns importance based on the number of links held by each node |
| **Eigenvector Centrality** | Assigns importance based on the number of links held by each node and on how well connected a node and its neighbours are |

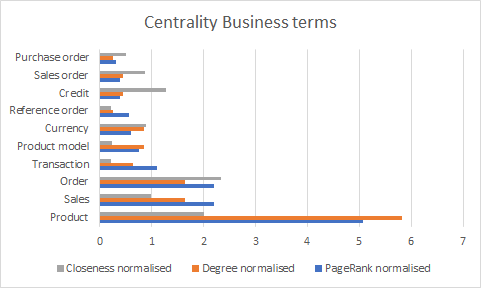
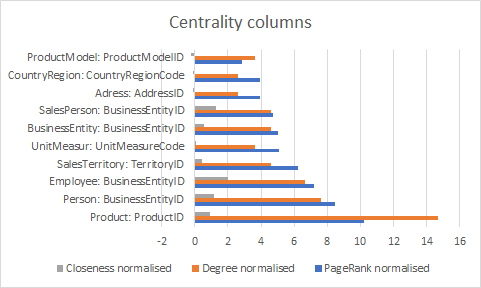
**Table 1** Definitions of measures

**Results**

Figures 8 and 9shows the results of the tests described above. The three measures are shown for the 10 business terms with the highest centrality. It can be seen that the term Product has both the highest degree centrality and the highest PageRank value, which means it is the term that is most referenced throughout our database of the Adventure Works company. Overall, degree centrality and PageRank centrality show a high correlation. Only for the terms ‘Product model’ and ‘Transaction’ is this not the case, indicating that although ‘Product model’ has more direct connections than ‘Transaction’ but the neighbors of ‘Transaction’ are more connected than the neighbors of ‘Product model’. In other words, the PageRank centrality measure gives more information about the indirect influence of the term. Because of this, it can also be seen that the PageRank measure attenuates the large differences in degree centrality between the most connected terms.

The same patterns can be seen when analyzing the centrality of the columns, with ‘ProductID’ being the node with the highest centrality. The column ‘BusinessEntityID’ occurs multiple times in figure 9, this is because this column occurs in various different tables. This high recurrence in addition to the high centrality of the individual columns makes standardization of its format extra important to maintain coherence in the database of the company.

The closeness centrality does not correlate well with the two other measures, this is no surprise since it tells us a different kind of story. More specifically, it provides us with a measure of how the node relates to the whole graph whereas the other two measures give more information about the local connectivity. Therefore, the closeness centrality does not give us much useful information when it comes to the ‘Business term’ and ‘Column’ nodes. However, when applied to the ‘User’ nodes, it gives us insight into which actor is most efficient in spreading information throughout the network. In this case, the employee with the last name ‘Hill’ has a much higher closeness value than her colleagues, indicating that she might be the most valuable person in the company when it comes to connecting the different data-silos or spreading new information.



**Figure 9**

**Figure 8**

# 5 Conclusion with focus on gained knowledge

For this assignment, we have reorganized the metadata from the Adventureworks database into a graph database. While some insights could have been extracted from the metadata itself, our main goal was to investigate potential applications of graph databases in a business environment and to discuss their added value. .

The first application explored is the monitoring of users in communities to ensure a proper adoption of metadata management software within an organization and use it as a tool to find new synergies between the departments. Secondly, we have described how graph database technology could be leveraged to perform impact analysis concerning the modification of an asset and the subsequent influence on the rest of the organization. Last but not least, we used various centrality algorithms to highlight the most important assets within the organizations. The latter is crucial as it could enable IT and management to better understand their importance and therefore integrate it when building or updating systems. Metadata management software does not only facilitate a better collaboration between the departments but also between business and IT. It also offers decision makers critical information and reveals blind spots in the company structure.

More use cases could have been integrated to this work such as reporting the incomplete number of assets per community or even incorporating additional node types within our model such as rules (OrderQty must be higher than 0) or reports. Once growing in complexity, such a model would be in measure to provide a transversal view on data taking the GDPR legislator's perspective as well as the one of the data scientist.

On top of the business insights gained during this assignment, we now have a better understanding of graph technology and of its advantages compared to traditional relational databases. Moreover, this assignment has enables us to ...

On top of the technical knowledge learned during the creation of the graph and our analysis, we better understand in which use cases it would be more beneficial to use graph database than relational database. Easier to build than a relational database, it can be gradually expanded with limited efforts. Moreover, although its implementation still required some skills, its usage is much more intuitive for business people as it can be directly used in some use cases without any technical skills.

# 6 Individual contribution:

## 6.1 François Wilberz

As a former consultant in data governance, I used to work with Collibra, a data governance software relying on graph databases. Through this assignment, I wanted to gain a deeper understanding of the functioning of such a technology while learning in parallel more technical insights about traditional relational databases. My contributions to this project are the following:

* Metadata extraction of the AdventureWorks database (SQL Server)
* Explanation to my teammates of the basic concepts of data governance
* Preparation of the CSV files to be read by Neo4J at the exception of the Business Term and the transformation data.
* Writing the queries to import the nodes and relationships information into Neo4J
* The redaction of the context, methodology and community overview section

## 6.2 Nicolas Delahousse

* Writing the section on Impact Analysis and gained knowledge
* Cleaning and preparing data for importation in Neo4J
* Performing the analysis on impact analysis

## 6.3 Robbe Neyns

* Preparation of the data for importation in Neo4J
* Performing the centrality analysis
* Writing the report on centrality analysis

# 7 Appendix

Appendix A: Definition of the different node type

* An **user** is an individual that have access to the platform.
* An **role** is the position that a user has within a community. Each role defined the permissions granted to the user having that role.
* A **responsibility** is the attribution of a role to an user for a specific community
* A **community** is a subgroup of the organization which is in charge of its own assets. The responsibilities are assigned at this level in this case to simply to keep think clear. In more elaborate schema, it could be assigned at the domain level (physical data dictionary or glossary) or even directly at the level of the data asset.
* A domain can be perceived as a folder containing only assets from a specific type. **Physical data dictionary** and **glossary** are different domain types. P**hysical data dictionary** will contain everything regarding physical data while the **glossary** should contain the **business terms**. In our disposition we limit each **community** to have only one **glossary** and one **physical data dictionary** but nothing prevent a **community** to have several business glossaries or physical data dictionaries.
* A **schema** is a logical division of the database containing a set of tables.
* A **table** is a set of columns organizing the data in a structured way. The instance of an asset type table can either be a physical table or a view.
* A **transformation** in our case is a query using in input several tables to create a view.
* A **column** is a vertical component of a table containing a set of data from the same type.

**Appendix B Extract evolution of number of assets per months**

**Match**

com:Community)-[:MANAGES]->(fol)-[]->(node)

WHERE node:Schema OR

node:Table OR

node:Transformation OR

node:Column OR

node:Business\_Term

WITH com.name AS Community\_name,

fol.name AS Domain\_name,

LABELS(node)[0] AS Node\_Type,

date(node.creation\_date).month AS Creation\_date\_month,

date(node.last\_modified\_date).month AS Last\_modified\_month,

count(node) As counting

RETURN DISTINCT Community\_name, Domain\_name, Node\_Type, Creation\_date\_month, Last\_modified\_month, counting

ORDER BY Community\_name,Domain\_name, Node\_Type, Creation\_date\_month, Last\_modified\_month

Appendix C: SQL queries to extract metadata from AdventureWorks in Microsoft SQL Server Management studio

Query to get all tables and views

SELECT \*

FROM INFORMATION\_SCHEMA.TABLES

**Query to get tables used to build views**

select distinct schema\_name(v.schema\_id) as schema\_name,

v.name as view\_name,

schema\_name(o.schema\_id) as referenced\_schema\_name,

o.name as referenced\_entity\_name,

o.type\_desc as entity\_type

from sys.views v

join sys.sql\_expression\_dependencies d

on d.referencing\_id = v.object\_id

and d.referenced\_id is not null

join sys.objects o

on o.object\_id = d.referenced\_id

order by schema\_name,

view\_name;

**Query to get row count**

USE [AdventureWorks2017]

GO

CREATE PROCEDURE dbo.ViewsRowCount

AS

BEGIN

SET NOCOUNT ON

CREATE TABLE #tempRowCount

(

Name VARCHAR(100),

Row\_Count INT

)

DECLARE @SQL VARCHAR(MAX)

SET @SQL = ''

SELECT @SQL = @SQL + 'INSERT INTO #tempRowCount SELECT ''' +

SCHEMA\_NAME(schema\_id) + '.' + name + ''', COUNT(\*) FROM ' +

SCHEMA\_NAME(schema\_id) + '.' + name +

CHAR(13) FROM sys.objects WHERE type = 'V'

EXEC (@SQL)

SELECT Name, Row\_Count

FROM #tempRowCount

END

GO

USE [AdventureWorks2017]

GO

EXEC dbo.ViewsRowCount

GO

**Queries to extract primary and foreign keys**

SELECT o2.name AS Referenced\_Table\_Name,

c2.name AS Referenced\_Column\_As\_FK,

o1.name AS Referencing\_Table\_Name,

c1.name AS Referencing\_Column\_Name,

s.name AS Constraint\_name

FROM sysforeignkeys fk

INNER JOIN sysobjects o1 ON fk.fkeyid = o1.id

INNER JOIN sysobjects o2 ON fk.rkeyid = o2.id

INNER JOIN syscolumns c1 ON c1.id = o1.id AND c1.colid = fk.fkey

INNER JOIN syscolumns c2 ON c2.id = o2.id AND c2.colid = fk.rkey

INNER JOIN sysobjects s ON fk.constid = s.id

ORDER BY o2.name

**Get all columns data information**

select \*

from sys.tables as tab

inner join sys.columns as col

on tab.object\_id = col.object\_id

left join sys.types as t

on col.user\_type\_id = t.user\_type\_id

Appendix D: Queries for nodes

**Query to load schema**

LOAD CSV WITH HEADERS FROM "file:///schema.csv" AS sch

FIELDTERMINATOR ';'

CREATE (schema:Schema {id: sch.id, name: sch.name, creation\_date: sch.creation\_date, last\_modified\_date: sch.last\_modified\_date})

**Query to load table**

LOAD CSV WITH HEADERS FROM "file:///table.csv" AS tab

FIELDTERMINATOR ';'

CREATE (table:Table {id: tab.id, name: tab.table\_name, type: tab.table\_type, row\_count: tab.row\_count, creation\_date: tab.creation\_date, last\_modified\_date: tab.last\_modified\_date})

**Query to load column**

LOAD CSV WITH HEADERS FROM "file:///column.csv" AS col

FIELDTERMINATOR ';'

CREATE (column:Column {id: col.id, name: col.column\_name, ordinal\_position: col.ordinal\_position, default\_value: col.column\_default, is\_nullable:col.is\_nullable, data\_type:col.data\_type,max\_length:col.character\_maximum\_length, numeric\_precision:col.numeric\_precision, primary\_key:col.primary\_key, composite\_primary\_key: col.composite\_primary\_key, creation\_date: col.creation\_date, last\_modified\_date: col.last\_modified\_date })

**Query to load business term**

LOAD CSV WITH HEADERS FROM "file:///Business\_term.csv" AS term

FIELDTERMINATOR ';'

CREATE (business\_term:Business\_Term {id: term.id, name: term.name, definition: term.definition, security\_classification: term.security\_classification, difficulty\_harmonization:term.difficulty\_harmonization, creation\_date: term.creation\_date, last\_modified\_date: term.last\_modified\_date })

**Query to load physical data dictionary**

LOAD CSV WITH HEADERS FROM "file:///physical\_dict.csv" AS dic

FIELDTERMINATOR ';'

CREATE (dictionnary:Physical\_data\_dictionary {id: dic.id, name: dic.name, creation\_date: dic.creation\_date, last\_modified\_date: dic.last\_modified\_date })

**Query to load glossary**

LOAD CSV WITH HEADERS FROM "file:///glossary.csv" AS glo

FIELDTERMINATOR ';'

CREATE (glossary:Glossary {id: glo.id, name: glo.name, creation\_date: glo.creation\_date, last\_modified\_date: glo.last\_modified\_date })

**Query to load community**

LOAD CSV WITH HEADERS FROM "file:///community.csv" AS com

FIELDTERMINATOR ';'

CREATE (community:Community {id: com.id, name: com.name, creation\_date: com.creation\_date, last\_modified\_date: com.last\_modified\_date})

**Query to load transformation**

LOAD CSV WITH HEADERS FROM "file:///transformation.csv" AS trans

FIELDTERMINATOR ';'

CREATE (transformation:Transformation {id: trans.id, name: trans.name, query: trans.Query, creation\_date: trans.creation\_date, last\_modified\_date: trans.last\_modified\_date })

**Query to load responsibility**

LOAD CSV WITH HEADERS FROM "file:///responsibility.csv" AS resp

FIELDTERMINATOR ';'

CREATE (responsibility:Responsibility {id: resp.id, name: resp.name,creation\_date: resp.creation\_date, last\_modified\_date: resp.last\_modified\_date})

**Query to load role**

LOAD CSV WITH HEADERS FROM "file:///roles.csv" AS rol

FIELDTERMINATOR ';'

CREATE (sys\_roles:Sys\_Roles {id: rol.id, name: rol.name,description: rol.description, creation\_date: rol.creation\_date, last\_modified\_date: rol.last\_modified\_date})

**Query to load user**

LOAD CSV WITH HEADERS FROM "file:///user.csv" AS use

FIELDTERMINATOR ';'

CREATE (sys\_users:Sys\_Users {id: use.id, first\_name: use.first\_name,last\_name: use.last\_name, sex: use.sex, email:use.email, phone: use.phone, creation\_date: use.creation\_date, last\_modified\_date: use.last\_modified\_date})

**Appendix E: QUERY FOR RELATIONSHIPS**

**Query to create relationship between schema and table**

LOAD CSV WITH HEADERS FROM "file:///table.csv" AS tab

FIELDTERMINATOR ';'

MATCH (p1:Schema {id:tab.schema\_id}), (p2:Table {id:tab.id})

CREATE (p1)-[:HAS]->(p2);

**Query to create relationship between table and schema**

LOAD CSV WITH HEADERS FROM "file:///table.csv" AS tab

FIELDTERMINATOR ';'

MATCH (p1:Schema {id:tab.schema\_id}), (p2:Table {id:tab.id})

CREATE (p2)-[:IS\_PART\_OF]->(p1);

**Query to create relationship between table and column**

LOAD CSV WITH HEADERS FROM "file:///column.csv" AS col

FIELDTERMINATOR ';'

MATCH (p1:Table {id:col.table\_id}), (p2:Column {id:col.id})

CREATE (p1)-[:IS\_COMPOSED\_OF]->(p2);

**Query to create relationship between column and table**

LOAD CSV WITH HEADERS FROM "file:///column.csv" AS col

FIELDTERMINATOR ';'

MATCH (p1:Table {id:col.table\_id}), (p2:Column {id:col.id})

CREATE (p2)-[:COMPOSES]->(p1);

**Query to create relationship between business term and column**

LOAD CSV WITH HEADERS FROM "file:///column.csv" AS col

FIELDTERMINATOR ';'

MATCH (p1:Business\_Term {id:col.term\_id}), (p2:Column {id:col.id})

CREATE (p1)-[:DEFINES]->(p2);

**Query to create relationship between column and business term**

LOAD CSV WITH HEADERS FROM "file:///column.csv" AS col

FIELDTERMINATOR ';'

MATCH (p1:Business\_Term {id:col.term\_id}), (p2:Column {id:col.id})

CREATE (p2)-[:IS\_DEFINED\_BY]->(p1);

**Query to create relationship between table and transformation**

LOAD CSV WITH HEADERS FROM "file:///transformation\_rel.csv" AS trans\_rel

FIELDTERMINATOR ';'

MATCH (p1:Table {id:trans\_rel.table\_id}), (p2:Transformation {id:trans\_rel.trans\_id})

CREATE (p1)-[:IS\_USED\_BY]->(p2);

**Query to create relationship between transformation and table**

LOAD CSV WITH HEADERS FROM "file:///transformation\_rel.csv" AS tran

FIELDTERMINATOR ';'

MATCH (t1:Table {id:tran.table\_id}), (t2:Transformation {id:tran. trans\_id })

CREATE (t2)-[:USES]->(t1);

**Query to create relationship between transformation and view**

LOAD CSV WITH HEADERS FROM "file:///transformation\_rel.csv" AS tran

FIELDTERMINATOR ';'

MATCH (t1:Table {id:tran.view\_id}), (t2:Transformation {id:tran.trans\_id})

WITH DISTINCT t1 AS t\_1,

t2

CREATE (t2)-[:GENERATES]->(t\_1);

**Query to create relationship between view and transformation**

LOAD CSV WITH HEADERS FROM "file:///transformation\_rel.csv" AS tran

FIELDTERMINATOR ';'

MATCH (t1:Table {id:tran.view\_id}), (t2:Transformation {id:tran.trans\_id})

WITH DISTINCT t1 AS t\_1,

t2

CREATE (t\_1)-[:IS\_GENERATED\_BY]->(t2);

**Query to create foreign key relationship between columns (is** FOREIGN KEY)

LOAD CSV WITH HEADERS FROM "file:///for\_key\_relationship.csv" AS for

FIELDTERMINATOR ';'

MATCH (t1:Column {id:for.column\_id1}), (t2:Column {id:for.column\_id2})

CREATE (t1)-[:IS\_FOREIGN\_KEY\_OF]->(t2);

**Query to create foreign key relationship between columns**

(HAS FOREIGN KEY)

LOAD CSV WITH HEADERS FROM "file:///for\_key\_relationship.csv" AS for

FIELDTERMINATOR ';'

MATCH (t1:Column {id:for.column\_id1}), (t2:Column {id:for.column\_id2})

CREATE (t2)-[:HAS\_FOREIGN\_KEY]->(t1);

**Query to create relationship between business term and glossary**

LOAD CSV WITH HEADERS FROM "file:///Business\_term.csv" AS term

FIELDTERMINATOR ';'

MATCH (t1:Business\_Term {id:term.id}), (t2:Glossary {id:term.glossary\_id})

CREATE (t1)-[:IS\_REGROUPED\_IN]->(t2);

**Query to create relationship between glossary and business term**

LOAD CSV WITH HEADERS FROM "file:///Business\_term.csv" AS term

FIELDTERMINATOR ';'

MATCH (t1:Business\_Term {id:term.id}), (t2:Glossary {id:term.glossary\_id})

CREATE (t2)-[:REGROUPS]->(t1);

**Query to create relationship between column and physical data dictionary**

LOAD CSV WITH HEADERS FROM "file:///column.csv" AS col

FIELDTERMINATOR ';'

MATCH (t1:Column {id:col.id}), (t2:Physical\_data\_dictionary {id:col.physical\_id})

CREATE (t1)-[:IS\_CONTAINED\_BY]->(t2);

**Query to create relationship between physical data dictionary and column**

LOAD CSV WITH HEADERS FROM "file:///column.csv" AS col

FIELDTERMINATOR ';'

MATCH (t1:Column {id:col.id}), (t2:Physical\_data\_dictionary {id:col.physical\_id})

CREATE (t2)-[:CONTAINS]->(t1);

**Query to create relationship between table and physical data dictionary**

LOAD CSV WITH HEADERS FROM "file:///table.csv" AS tab

FIELDTERMINATOR ';'

MATCH (t1:Table {id:tab.id}), (t2:Physical\_data\_dictionary {id:tab.physical\_id})

CREATE (t1)-[:IS\_CONTAINED\_BY]->(t2);

**Query to create relationship between physical data dictionary and table**

LOAD CSV WITH HEADERS FROM "file:///table.csv" AS tab

FIELDTERMINATOR ';'

MATCH (t1:Table {id:tab.id}), (t2:Physical\_data\_dictionary {id:tab.physical\_id})

CREATE (t2)-[:CONTAINS]->(t1);

**Query to create relationship between schema and physical data dictionary**

LOAD CSV WITH HEADERS FROM "file:///schema.csv" AS sch

FIELDTERMINATOR ';'

MATCH (t1:Schema {id:sch.id}), (t2:Physical\_data\_dictionary {id:sch.physical\_id})

CREATE (t1)-[:IS\_CONTAINED\_BY]->(t2);

**Query to create relationship between transformation and physical data dictionary**

LOAD CSV WITH HEADERS FROM "file:///transformation.csv" AS trans

FIELDTERMINATOR ';'

MATCH (t1:Transformation {id: trans.id}), (t2:Physical\_data\_dictionary {id: trans.physical\_id})

CREATE (t1)-[:IS\_CONTAINED\_BY]->(t2);

**Query to create relationship between physical data dictionary and transformation**

LOAD CSV WITH HEADERS FROM "file:///transformation.csv" AS trans

FIELDTERMINATOR ';'

MATCH (t1:Transformation {id: trans.id}), (t2:Physical\_data\_dictionary {id: trans.physical\_id})

CREATE (t2)-[:CONTAINS]->(t1);

**Query to create relationship between physical data dictionary and schema**

LOAD CSV WITH HEADERS FROM "file:///schema.csv" AS sch

FIELDTERMINATOR ';'

MATCH (t1:Schema {id:sch.id}), (t2:Physical\_data\_dictionary {id:sch.physical\_id})

CREATE (t2)-[:CONTAINS]->(t1);

**Query to create relationship between community and physical data dictionary**

LOAD CSV WITH HEADERS FROM "file:///Community.csv" AS com

FIELDTERMINATOR ';'

MATCH (t1:Community {id:com.id}), (t2:Physical\_data\_dictionary {id:com.physical\_id})

CREATE (t1)-[:MANAGES]->(t2);

**Query to create relationship between physical data dictionary and community**

LOAD CSV WITH HEADERS FROM "file:///Community.csv" AS com

FIELDTERMINATOR ';'

MATCH (t1:Community {id:com.id}), (t2:Physical\_data\_dictionary {id:com.physical\_id})

CREATE (t2)-[:IS\_MANAGED\_BY]->(t1);

**Query to create relationship between community and glossary**

LOAD CSV WITH HEADERS FROM "file:///Community.csv" AS com

FIELDTERMINATOR ';'

MATCH (t1:Community {id:com.id}), (t2:Glossary {id:com.glossary\_id})

CREATE (t1)-[:MANAGES]->(t2);

**Query to create relationship between glossary and community**

LOAD CSV WITH HEADERS FROM "file:///Community.csv" AS com

FIELDTERMINATOR ';'

MATCH (t1:Community {id:com.id}), (t2:Glossary {id:com.glossary\_id})

CREATE (t2)-[:IS\_MANAGED\_BY]->(t1);

**Query to create relationship between role and responsibility**

LOAD CSV WITH HEADERS FROM "file:///responsibility.csv" AS resp

FIELDTERMINATOR ';'

MATCH (t1: Responsibility {id:resp.id}), (t2: Sys\_Roles {id:resp.role\_id})

CREATE (t2)-[:DESCRIBES]->(t1);

**Query to create relationship between responsibility and role**

LOAD CSV WITH HEADERS FROM "file:///responsibility.csv" AS resp

FIELDTERMINATOR ';'

MATCH (t1: Responsibility {id:resp.id}), (t2: Sys\_Roles {id:resp.role\_id})

CREATE (t1)-[:IS\_DESCRIBED\_BY]->(t2);

**Query to create relationship between user and responsibility**

LOAD CSV WITH HEADERS FROM "file:///responsibility.csv" AS resp

FIELDTERMINATOR ';'

MATCH (t1: Responsibility {id:resp.id}), (t2: Sys\_Users{id:resp.user\_id})

CREATE (t2)-[:IS\_INVOLVED\_IN]->(t1);

**Query to create relationship between responsibility and user**

LOAD CSV WITH HEADERS FROM "file:///responsibility.csv" AS resp

FIELDTERMINATOR ';'

MATCH (t1: Responsibility {id:resp.id}), (t2: Sys\_Users {id:resp.user\_id})

CREATE (t1)-[:INVOLVES]->(t2);

**Query to create relationship between community and responsibility**

LOAD CSV WITH HEADERS FROM "file:///responsibility.csv" AS resp

FIELDTERMINATOR ';'

MATCH (t1: Responsibility {id:resp.id}), (t2: Community{id:resp.community\_id})

CREATE (t2)-[:IS\_ADMINISTERED\_BY]->(t1);

**Query to create relationship between responsibility and community**

LOAD CSV WITH HEADERS FROM "file:///responsibility.csv" AS resp

FIELDTERMINATOR ';'

MATCH (t1: Responsibility {id:resp.id}), (t2: Community {id:resp.community\_id})

CREATE (t1)-[:ADMINISTERS]->(t2);

Appendix F: Community queries

Query to highlight users having responsibilities in more than 2 communities

MATCH (use:Sys\_Users)-[:IS\_INVOLVED\_IN]->(resp:Responsibility)-[:ADMINISTERS]->(com:Community)

WITH use.last\_name AS Last\_name,

use.first\_name AS First\_name,

collect(DISTINCT com.name) AS LIST\_NAME

WHERE size(LIST\_NAME)>1

WITH Last\_name,

First\_name,

LIST\_NAME

MATCH (use:Sys\_Users)-[:IS\_INVOLVED\_IN]->(resp:Responsibility)-[:IS\_DESCRIBED\_BY]->(rol:Sys\_Roles), (com:Community)-[:IS\_ADMINISTERED\_BY]->(resp:Responsibility)

WHERE use.last\_name = Last\_name AND use.first\_name = First\_name

RETURN use.last\_name AS Last\_Name, use.first\_name AS First\_Name, rol.name AS Position, com.name AS Community\_Name

ORDER BY use.last\_name

Appendix G: Queries for impact analysis

**Query to get the impact of a change in account definition on the sales community**MATCH (c:Community)-[:MANAGES]->(pg:Physical\_data\_dictionary)  
MATCH (c:Community)-[:MANAGES]-(gl:Glossary)  
MATCH (gl:Glossary)--(bt:Business\_Term {name:"Account"})  
MATCH (pg:Physical\_data\_dictionary)-[:CONTAINS]-(t:Table)  
MATCH (tr:Transformation)--(t:Table)-[:HAS]-(col:Column)  
MATCH (col:Column)--(bt:Business\_Term)  
RETURN \*

**Query to get the impact of a change in account definition on the sales community and purchasing community**  
MATCH (col:Column)--(bt:Business\_Term {name:"Account"})  
MATCH (c:Community)-[:MANAGES]->(pg:Physical\_data\_dictionary)  
MATCH (tr:Transformation)--(t:Table)-[:HAS]-(col:Column)  
WHERE c.name = 'Purchasing community' or c.name = 'Sales community'  
RETURN \*

**Query to find the responsibilities of users in both the purchasing and sales community**  
MATCH (col:Column)--(bt:Business\_Term {name:"Account"})  
MATCH (c:Community)-[:MANAGES]->(pg:Physical\_data\_dictionary)  
MATCH (c:Community)--(r:Responsibility)--(sys:Sys\_Users)  
MATCH (c:Community)--(r:Responsibility)--(sr:Sys\_Roles)  
MATCH (col:Column)--(t:Table)--(q:Transformation)  
WHERE c.name = 'Purchasing community' or c.name = 'Sales community'  
RETURN c,r,sr,sys