

**Identify Multi-level Hierarchies and Tree Structure of
Group Chain Companies using Graph data and Relational Databases**

By

Bijen Shah – W1918439

MSc Data Science and Analytics – Sept 2022-23

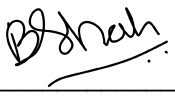
Supervised by

Dr Panagiotis Chountas

Declaration

I, Bijen Shah, declares that I am the sole author of this Project; that all references cited have been consulted; that I have conducted all work of which this is a record, and that the finished work lies within the prescribed word limits.

This has not previously been accepted as part of any other degree submission.

Sign:  _____

Date: 05/09/2023

Table of Contents

Acknowledgements	4
Abstract	5
Table of Tables	6
Table of Figures.....	6
1. Introduction	7
1.1. Understanding Problem	7
1.2. Aim	8
1.3. Objectives.....	8
1.4. Existing solutions	9
1.5. Defining Attributes	9
1.5.1. Beneficiaries.....	9
1.5.2. Ownership	10
1.5.3. SIC	10
1.5.4. RTIC	11
1.5.5. Users	11
1.5.6. Use Cases.....	12
1.5.7. Why ownership information matters?.....	12
1.5.8. Potential use of Group Structures on the platform.....	12
1.5.9. Limitations of current analysis and insights	12
1.6. Areas of Data	13
1.6.1. Open Ownership Data.....	13
1.6.2. Beneficial Ownership Data Standard (BODS)	14
1.6.3. Linkurious	15
1.6.4. Companies House.....	15
1.6.5. The Data City	16
1.7. Challenges	16
1.7.1. Availability of data	20
1.7.2. Visualization Data.....	20
1.7.3. Graph & Relational Database limitations	22
1.7.4. Data Limitations	23

1.8.	Development Methods	23
1.8.1.	How to find Company Ownership Chains?	23
1.8.2.	Why recursion?	25
1.8.3.	What other Group Structures to find?	27
2.	Project Management.....	27
2.1.	Overview	27
2.2.	Time management and sprints.....	28
2.3.	Kanban	28
2.4.	Integration with The Data City.....	29
3.	Architecture, Design, and Implementation.....	29
3.1.	Overview	29
3.2.	Data Extraction.....	29
3.3.	Choosing data source types and granularity.....	30
3.4.	Data Storing	30
3.5.	Data Indexing	30
3.5.1.	Index and Query Performance	30
3.6.	Data transformation layers	31
3.7.	Merging relevant data and ETL orchestration design	32
3.8.	Data Output and Further Integration	33
3.9.	Language	33
3.9.1.	SQL	34
3.9.2.	Python	34
3.10.	Database Software.....	35
3.11.	Algorithms and Functions.....	35
3.12.	Setup & Installation	35
4.	Use Case of Analysed Data on RTIC Lists	36
4.1.	Use case problem.	36
4.2.	Confusion for end user.....	36
4.3.	Solution provided.	36
4.4.	Time saving	37
4.5.	Analysis Potential	38

5. GitHub Version control	38
5.1. GitHub	38
5.2. How version control helped in the project?	39
6. Future Scope	40
6.1. Potential	40
6.2. Project Success and Motivation	41
6.3. Limitations	41
7. Conclusion	43
References	44

Acknowledgements

I would like to thank my parents and my sisters without whom none of this would be possible. They have sacrificed and invested so much for me; I will be forever grateful of them. I would also like thank my brother for being a great support whilst completing his masters' studies with me and making the struggle happening.

I would like to thank Jack Lewis for giving me the chance to work with The Data City for my master's dissertation project.

I would like to thank Professor Panagiotis Chountas for helping me through my masters' studies and making the journey worthwhile.

Finally, I would like to thank my Kalyan Mitras for encouraging and motivating me.

Abstract

The project will be completed using The Data City's (TDC) data of UK registered companies. This industry project will involve my collaboration with the TDC team, as we aim to understand how to classify data from UK registered companies using data engineering rules (objectively) and/or data science tools (subjectively). The goal is to overcome the limitations of Standard Industrial Classifications (SICs). For each company listed on Companies House, we intend to deduce the group tree structure by leveraging both shareholder information and details on Persons with Significant Control (PSIC). Using both Open Ownership Data and TDC data we will try to identify company group tree structures and company chains using the relations and hierarchy between companies. The identification of search tree structures will allow TDC's platform to better visualise information of companies and help stake holders and users make better decisions as they will understand the fundamentals of these group companies better. Example of Tesco PLC having 350K+ employees and Tesco Stores Limited having 240K+ employees. The idea is to use the shareholders and company group/chain structure information and identify the distinction between the company's employee count.

The identification of the Real Time Industry Classification (RTIC) using Open Ownership graph data and relational database like SQLite of Companies and Group of Businesses will allow TDC to deliver accurate KPI and fundamentals on their platform to their customers.

Table of Tables

Table 1: Flags and Definitions in Use Case Problem.....	37
---	----

Table of Figures

Figure 1: The Data City Product	8
Figure 2: Ownership example of Open Ownership Register	10
Figure 3: Company Relationships found in Linkurious	13
Figure 4: Beneficial Ownership Data Standard.....	14
Figure 5: Companies House Information about Companies.....	16
Figure 6: Companies House Information about Companies.....	17
Figure 7: Hierarchy and Level of Companies in a Group Structure	18
Figure 8: Line graph showing count of Companies (100Ks) for Hierarchy Levels	18
Figure 9: Investigating different group structures in Linkurious Platform.....	19
Figure 10: Top 5 Ultimate Parent Companies based on total count of sub companies.	21
Figure 11: Top 6th-10th Ultimate Parent Companies based on total count of sub companies.	21
Figure 12: Top 10 Estimated Group Companies based on total count of companies and beneficiaries	22
Figure 13: Company Group Structures as defined in our Data Tables	22
Figure 14: Stage 1 finding of Company Chains	23
Figure 15: Stage 2 of finding Company Chains	24
Figure 16: Stage 3 Finding Ultimate Parent Companies	24
Figure 17: Final Stage Finding all sub companies using SQL recursion	25
Figure 18: Visualising Companies in group chains with companies above and below in the hierarchy	26
Figure 19: Companies in group chains starting from U - Ultimate Parent through P - Parent Companies and to C - Child Companies.....	26
Figure 20: List of tasks listed on Kanban Dashboard for tracking development.....	27
Figure 21: Kanban Chart of Project Tasks	28
Figure 22: Extract Transform and Load Process of Project	29
Figure 23: The Data City Product - Company Group Details.....	33
Figure 24: Data and Flags Updated of Use Case Problem	37
Figure 25: Github Repository of the Project.....	39
Figure 26: The Data City Global Product (Beta)	41

1. Introduction

This is an industry project done in collaboration with The Data City (TDC). The company's mission is to map the UK's emerging economy, providing researchers, policy makers and investors with real time data on dynamic sectors and the companies within them. TDC provides a product which allows users to view information like financials, employment, growth, industry, etc. about companies. The users use this data in their daily work in defining ill-defined sectors using traditional datasets (UK Government), finding sales leads, investment opportunities and other opportunities. The companies listed in different jurisdictions throughout the country lacked information about company group structure unless they want the public to know. TDC's product classifies these companies based on their properties and their team wants to group companies based on their beneficiary company and group structures. At the moment, they only have limited information on group structure and struggle with cascading a whole group structure into one parent company.

1.1. Understanding Problem

TDC's product extracts data from the internet for each company on Companies House and finds relevant information and insights by matching and scraping their website. Their IP is classification of businesses. The data is then visualised on the platform as shown in the below snapshot (Fig 1). The problem that TDC users have is that they see a lot of companies which are either sub companies of the same chain/group of companies or are sub companies created for accounting purposes. The user when using the portal needs to know which company belongs to which group of companies and rather if there could be a group of companies by understanding the beneficiaries. This information helps the users save time to make more calculated decisions when tackling business decisions and or taking time out to contact these companies for future opportunities.

Since, the companies listed around the world only usually have the information about its immediate benefitting company and/or beneficiaries. This information is not enough when planning on vast company groups. Hence, the users want to group all this lists into groups of companies that they belong. To make the problem fundamentally simple, users want to see the Ultimate Beneficiary company and then other companies in that group collapsed underneath. For example, in the screenshot above we see two companies having the same website, but they might be part of a whole different group of companies. This way if we know which company belongs to which company group we can decide on our granularity and plan our business tasks accordingly.

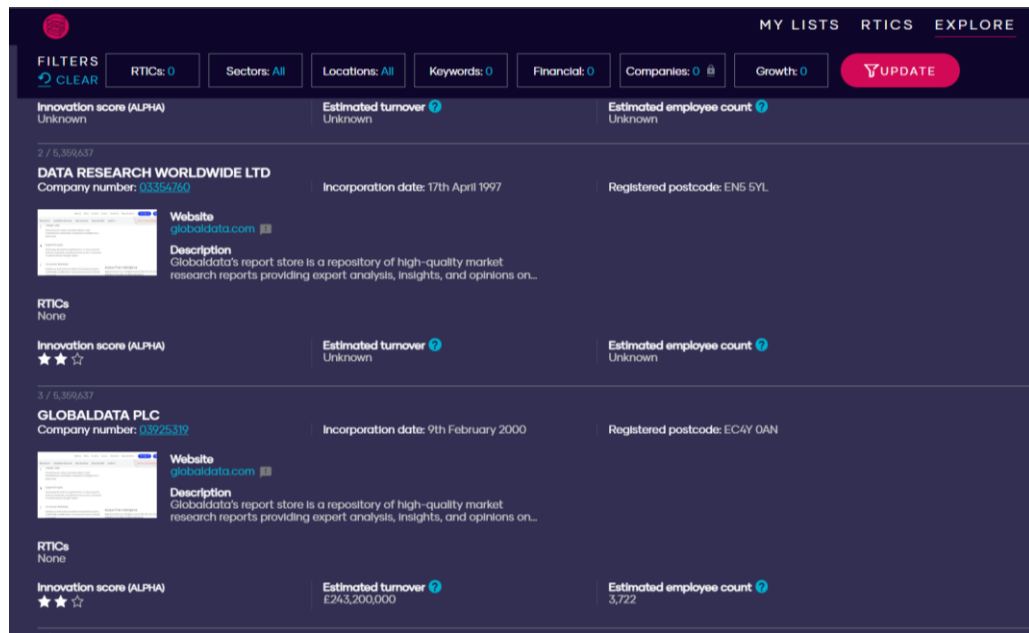


Figure 1: The Data City Product

Source: (The Data City, 2023)

1.2. Aim

Beneficiaries' information is one of the most important information when discussing about companies and their financials. This way the user understands who benefits from the business and whom to be contacted for business opportunities. TDC product has pioneered gathering all this information from the web and visualizing the information using machine learning algorithms and data engineering to its best. The beneficiary's information extraction will help enhance the TDC product fundamentally. The aim is to build an orchestration that can be continuously improved and developed for better beneficiaries' insights using graph data, relational databases, recursion methodologies and flattening the relationships of companies on a simple understandable table format.

1.3. Objectives

The project has the following objectives and outcomes:

1. Identify group of companies and their ultimate parent company (entities beneficiaries).
2. Estimate group of companies based on company details for person beneficiaries.
3. Use the insights on TDC product Real Time Industry Classification lists for bucketing companies.
4. Build an infrastructure to simplify graph data into a table on a relational database.

5. Orchestration pipeline to be built for the data transformations.
6. Develop a process to use this information on a list for user acceptance testing and validation of bucketing of companies.

1.4. Existing solutions

The Data City has some existing solution to derive company structures, but this is not based on the Open Ownership dataset used in this project. The data is sourced from a credit rating company and is not scaled for all the company registration jurisdictions who have their beneficiary's data available to the public. Thus, trying to orchestrate the data in house and be able to update the information whenever possible and required is one of the motivating factors of the project.

1.5. Defining Attributes

The project discusses a lot of different attributes which are explained below.

1.5.1. Beneficiaries

Beneficial ownership means those who ultimately own or control an asset, for example, a property or company. It is useful to know who the beneficial owner(s) of corporate structures are, as the beneficial owner(s) may be different from the legal owner(s). (Economic Crime and Corporate Transparency Bill 2022: Factsheets, 2023). For the project the beneficiaries are understood as a person or entity which owns a company (entity) and is listed in the interested party statement of the Open Ownership dataset.

Using Open Ownership data to define a beneficiary of a company:

Ownership statements in Open Ownership data have extensive details about the ownership of a company. Each statement gives the following information.

1. `subject_describedByEntityStatement` : The subject company whose ownership is discussed in the statement.
2. `interestedParty_describedByEntityStatement`: The interested party – company beneficiary who has some ownership over the subject company.
3. `interestedParty_describedByPersonStatement`: The interested party – person beneficiary who has some ownership over the subject company.

These are the three key features we use from the ownership statement to identify and define a beneficiary of a company.

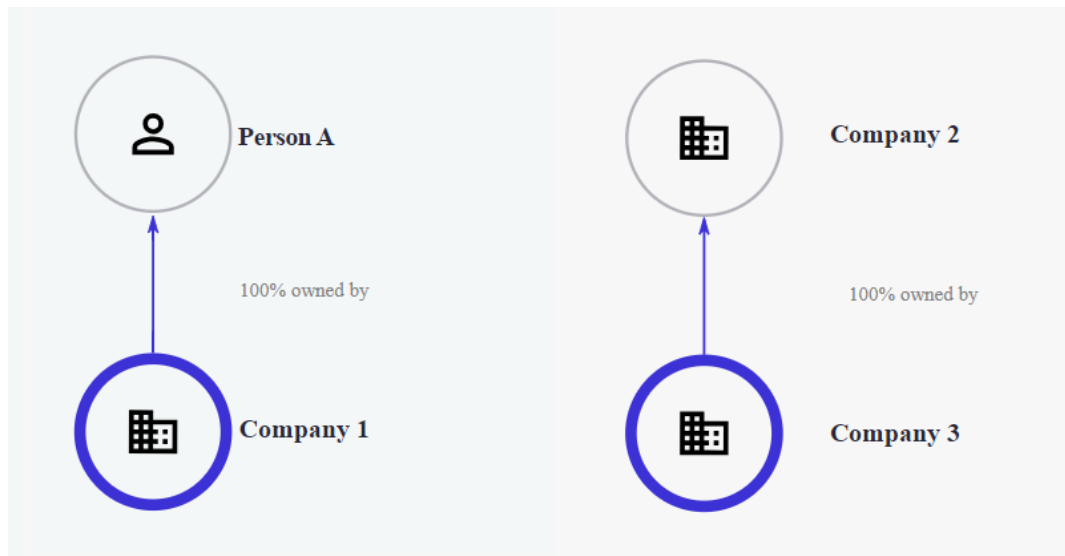


Figure 2: Ownership example of Open Ownership Register

(Open Ownership Register, 2023)

1.5.2. Ownership

Deciding how many entities or person can have ownership over each company was critical. This would become the base of the project and based on the decision made here the insights derived might be valuable to the project or be completely misleading. For each entity when there were more than one entity or person beneficiary it would make it difficult to derive chains and assign a single ultimate parent beneficiary. Hence, all beneficiaries are considered as owners of the entity. The chain and ultimate parent identification is built to cater to this 1 to many combinations and thus giving us multiple ultimate parents for some entities. Each ultimate parent gives us the whole chain of companies it owns.

Using Open Ownership data to define an ownership of a company. Using the beneficiaries identified in the above steps we can then use other details:

1. `publicationDetails_publicationDate`: This allows us to identify when the ownership was published.
2. Joining the information with the ownership statement interest data to find even more details on the statement. Type of ownership, ownership start and end date, etc. allows us to confirm the list of owners that a company has excluding all owners in the history.

1.5.3. SIC

The Standard Industrial Classification (SIC) was a system for classifying industries by a four-digit code as a method of standardizing industry classification for statistical purposes across agencies. Established in the United States in 1937, it is used by government agencies to

classify industry areas. Similar SIC systems are also used by United Kingdom's Companies House. (Standard Industrial Classification, 2023)

Interestingly, the project problem is aimed to overcome the classification in SIC by enhancing the Real Time Industry Classification (RTIC) built by The Data City using the SIC. In the project I am using the SIC codes for each entity and then deriving the chains or groups of companies further utilising this information on the RTIC list to enhance its visual by grouping companies who are part of the same group or have the same Ultimate Parent entity based on their SIC codes.

1.5.4. RTIC

RTICs provide a comprehensive and accurate picture of the economy, going beyond the limitations of traditional systems such as SIC codes. The Data City's innovative platform, featuring over 5 million companies, uses a combination of machine learning and expert training to determine what companies do and keep RTICs up to date with the latest advancements and innovations in the sector. (RTIC, 2023) From the project point of view, an RTIC is just a list of companies whose fundamentals match for the set. The companies listed in the RTIC need to be grouped based on their relevant group of companies.

1.5.5. Users

The user is defined as anyone who uses The Data City product for their analysis and tries to understand the RTICs, list of companies, looks for companies' information, ownership statuses, financials, etc. Different types of users have different use cases and meanings of this companies' details and relational data. For example, a user from a private equity investment company uses the product to find new companies, sectors, RTICs, etc. to invest their funds and create business opportunities. Another user from marketing uses the list of companies and their relations to create new leads for their business, marketing campaigns, research, etc. A user who is actively involved in recruiting, learning, and developing new teams may want to search more on companies who have similar businesses and growth potential. For a user looking at the data to find more growth opportunities in the UK this data could be the source of their analysis on how well few sectors are performing in some locations and how centred sectors are based on their geography. Any user who looks at the data and can have meaningful use of it to help their daily work and business benefits from additional information on company relations, group structures and ultimate parent company information.

1.5.6. Use Cases

Specific use cases are discussed later in the project to use the group structures on the custom made RTICs and evaluate how many children companies will be collapsed with the companies group information. This helps users make fast and efficient decisions and not invest time and resources contacting the child companies whom they might not want to contact for the best of their business interests. A different python notebook and process is created to use the company structures evaluated on the custom RTICs and create a group based on it.

1.5.7. Why ownership information matters?

Any human looking at a company's financials or doing business with them or invest in them or do any sort of work with them would like to know who benefits from that company and who owns that company. This gives the user/human a sense of understanding of knowing the company better. Similarly, when looking at a list of companies in The Data City's RTICs a user needs to know the immediate owner and the ultimate owner of that company. This allows the users to make better decisions, save time, focus on relevant information, and gain valuable insights when looking at financials. (Uehlinger, 2023)

1.5.8. Potential use of Group Structures on the platform

The aim is to potentially use the company group structures on the platform and create a collapsible user interface where all the child companies are listed under the ultimate parent available in that list of RTIC or view of companies investigated. This interface will allow the user to focus on the total number of groups of companies in the RTIC and then deep dive into each group as and when needed.

1.5.9. Limitations of current analysis and insights

A very interesting use of Open Ownership Registered data was found while investigating resources and tools which could be useful for the project. A company called Linkurious uses the same Open Ownership dataset and has created graph insights platform which allowed us to research on different types of group structures and chains of companies. An example is discussed below:

1. Where a group of companies owned by two beneficiaries Mr Mark Anthony Lamb and Mr Scott Anthony Daniels is shown.
2. It is interesting to see that company called Daniels Holdings Ltd. Is registered at the same address of the two beneficiaries.

3. Daniels Holdings Ltd. is known to be owned by another two beneficiaries Mr Scott Daniels and Avenue & Partners Ltd (further owned by MARK LAMB)

The insight from such a confusing group structure is that Mr Daniels and Mr Lamb own all of these companies in one or other way. Their ownership % might be different but eventually they benefit from the businesses. Although their names as understood by the jurisdictions are different in the two groups and hence are considered as two different person beneficiaries. This is one of the limitations of the Open Ownership dataset we might see for now. When our users who know about Scott and Marks owned companies might get confused looking at the group structures we evaluate because it might not group all of these into 1 group of companies.

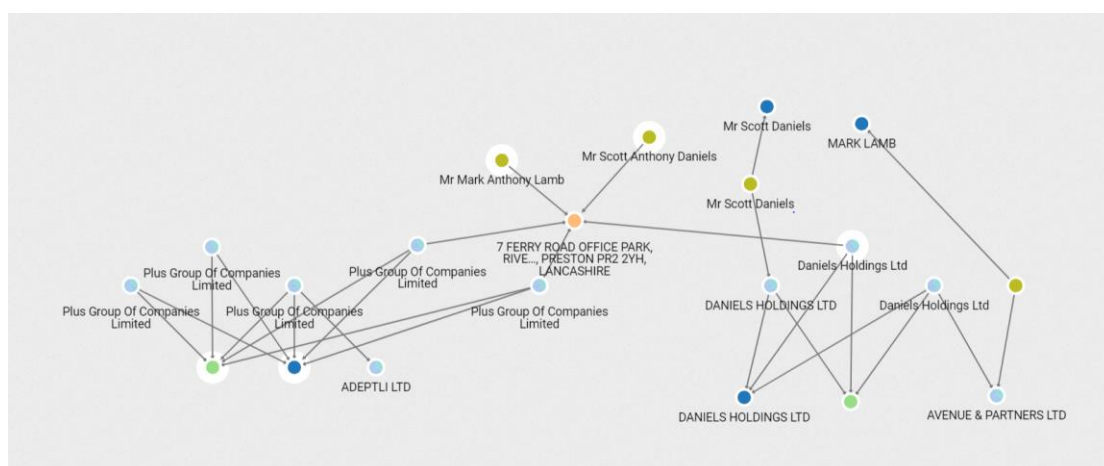


Figure 3: Company Relationships found in Linkurious

(Linkurious, 2023)

1.6. Areas of Data

1.6.1. Open Ownership Data

Each file contains a complete snapshot of all of the data in this register. That includes data from UK People with significant control, Denmark Central Business Register and Slovakia Public Sector Partners Register. The raw data from each of these sources has been imported, the companies reconciled with Open Corporates and - where possible - de-duplicated based on Open Corporates results and our own de-duplication algorithms. It has also been periodically refreshed with reconciliations with Open Corporates database to collect company name changes, dissolutions, etc.

The data is formatted in version 0.2 of Open Ownership's Beneficial Ownership Data Standard (BODS). All source data has been standardised on export, attempting to keep as much source-specific information as possible, but prioritising consistency across sources.

The file is in JSON Lines format, meaning each BODS statement is a JSON object on a single line of the file. Regularly data updates at least every month, and they will make a new export after each update. (Open Ownership Register, 2023)

From the project use this data is the base of the project. Insights drawn out of this data and group structures defined based on this data are to be utilized on the platform. The accuracy of this Open Ownership register directly impacts the accuracy of the insights and findings of the analysis.

1.6.2. Beneficial Ownership Data Standard (BODS)

BODS format provides a data modelling concept to encapsulate information about companies, their beneficiaries, and their respective details. This standard captures direct and indirect relationship between beneficiaries and companies. The standard also allows for the analysis to follow a certain defined framework while evaluating owners and beneficiaries. Standards in future if improved or amended will also allow for the project to change its logics according to the standard. An example of the BODS standard is shown below. Here a statement_ID defines a person identity with its details and another statement_ID defines the entity (Renco Energy). An ownership statement_ID is created to connect to show the relationship between the interested party (person or entity) and the subject entity whose ownership is being discussed.

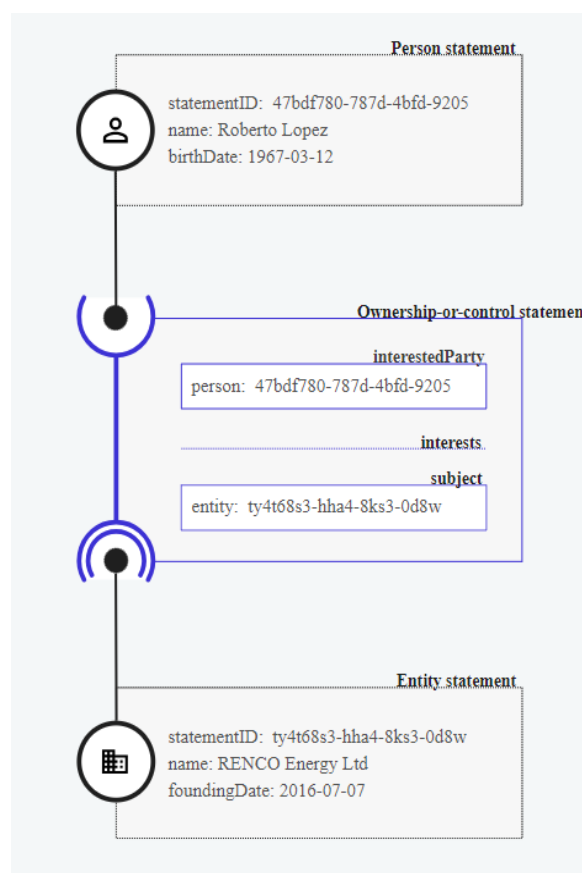


Figure 4: Beneficial Ownership Data Standard

(Beneficial Ownership Data Analysis Tool, 2023)

When the information is available in a structured fashion, its usefulness is increased. This improves the data's ability to reveal international networks of illicit financial flows and enable efficient and timely due diligence by making it simple to examine and correlate the data with other datasets. By offering a strong conceptual and practical framework for gathering and publishing beneficial ownership data, BODS facilitates beneficial ownership transparency and produces data that is interoperable, more readily reused, and of higher quality. In collaboration with Open Data Services, BODS is being developed by Open Ownership, a non-profit

organisation devoted to beneficial ownership transparency. Additionally, assistance and guidance are provided by an open data standard working group composed of data experts, experts in beneficial ownership, and other interested parties.

1.6.3. Linkurious

The visualisation that we see about the BODS format is not enough for us to create groups structures. I required the ability to visualise relationships between entities and persons based on their relative details for example their address, date of ownership, etc. Whilst researching more on the problem I found out about a company called Linkurious who uses the Open Ownership data set and allows the users to visualise the relationship between entities and persons and expand the graphical use of data to their respective details for example address, date of birth, date of ownership, date of creation, etc. The Linkurious platform allowed me to visualise relationship patterns between group structures which were not found through company group structures or ownership data (SIC coded relations). For example, a group of company which is owned by a single person, but the entities do not have a link between them these kind of relationship patterns were identified using Linkurious platform. Then a custom query tailored to that pattern was run on the whole of Open Ownership dataset to identify how many other groups of companies follow the same pattern.

1.6.4. Companies House

Companies House is an executive agency sponsored by the government of UK. At their core, they manage incorporating and dissolving companies. Also, they creating a register of all of these companies and make it available to the public. Companies House became the validation point for the project. After getting insights over company group structures and patterns, Companies House data was used to validate the structures derived and also validate information used from Open Ownership.

Although the research is not only based on the companies listed in the UK, but it was interesting to validate the group structures which were found in the UK jurisdiction using the Companies House data. When talking about the limitations of Open Ownership data and the overall project it is important to understand that some limitations were directly impacted because of Companies Houses data collection standards. If a registered person or entity at company's house cannot have a duplicate entry with different name structures that would help the analysis be more accurate but again this practise has to be made in place for all other jurisdictions or agencies like Companies House.

Search for a company or officer

Q

[Advanced company search](#)

TESCO PLC

Company number **00445790**

Follow this company

File for this company

Overview

Filing history

People

Charges

More

Officers

Persons with significant control

Filter officers

☐

Current officers

69 officers / 58 resignations

Figure 5: Companies House Information about Companies

(Companies House, 2023)

1.6.5. The Data City

1.7. Challenges

During the project I faced a few hurdles which were discussed briefly with the team at The Data City. This helped me also collaborate with them and understand about the project's expected outcome and utilization. I break them down into technical and fundamental challenges.

Technical Challenges:

1. Starting with the project we only had information about the BDOS standard and how each statement of ownership is a JSON statement. This was a major issue at the start of the project because we wanted to store this information into a relational database and flatten the relationship between the companies and query patterns out of it. Transforming the JSON file effectively and planning to run the same transformation with the data updates at Open Ownership monthly would be a difficult task because transformation of JSON using python was initially happening with a dozen of hardcoded logic and error handling. To overcome this, I had to further research on the Open Ownership platform to discover the Open Ownership register (Beneficial Ownership Data Analysis Tool, 2023). This tool was game changing for the project

in the initial stage. With this tool I could use the data in different formats viz. CSV, JSON, SQLite, PostgreSQL, Parquet, Big Query, etc. Now it was easier to start querying the data and understanding the relations, statements, persons, entities, etc.

2. Another technical challenge was querying this huge dataset on an SQLite dB and query relations with quick results and high computing efficiency. Although I was using a virtual desktop capable enough to query this huge data, the tool we choose to view this SQLite dB data was DB Browser (DB Browser for SQLite, 2023). The tool needed to be configured for the best query performance, transformation, and load efficiency, creating indexes, etc.

To achieve this

- a. I had to further go through of documentation on the dB browser webpage.
- b. Discuss with other users of dB browser.
- c. Get knowledge of best practices suggested by dB browser users.
- d. Find articles on similar issues and remedies suggested by dB browser users.

There was no one solution to this challenge. With the nature of different queries, I ran over the data I had to tweak the dB browser to run accordingly. (sqlitebrowser, 2023) One interesting insight I learnt while looking for a solution was that dB browser was basically a browser using SQLite configurations. This directed me to finding database administration best practices of SQLite. (Pragma Statements, 2023)

The solutions that were found around optimizing the pragma statements and utilizing the best out of the SQLite functionality drastically changed the time taken to find insights (Extraction) making changes to the available data (Transformation) and storing the insights into multiple tables (Loading) (Phiresky's Blog, 2020). It was interesting to see how the optimisations done would affect the database storing, logging, and journaling methodologies. At times the insights gathered were lost because of inappropriate optimisation as the database got locked while performing (Pragma Statements, 2023).

3. Recursively querying the relationship between companies and defining limitations to it was another challenge which had moreover a fundamental solution to it. When looking at companies and their ownership patterns one can imagine how a chain of ownership can circle back to the same company and make a circle of ownership pattern. The circular ownership patterns were a bottleneck for the recursive query built to identify chains and groups of companies.



Figure 6: Companies House Information about Companies

Since, the SQLite query wouldn't know if it was part of an infinite query loop and would indefinitely keep on looping over the same set of data. To overcome this challenge, I had to find out how many levels of hierarchy I wanted to investigate when finding relationships. After discussing this challenge with The Data City team, we concluded to restrict the recursive query

to a certain amount of relationship hierarchy and company level in the group. This is again a limitation to the technical capability of the performance of the SQLite database querying. It is possible to increase this limit if needed but essentially then number of insights as we increase the limit do get saturated and diminished after a certain number. To visualise this, I had to count the number of groups of companies and their maximum hierarchy level. (Optimizing a recursive CTE , 2021)

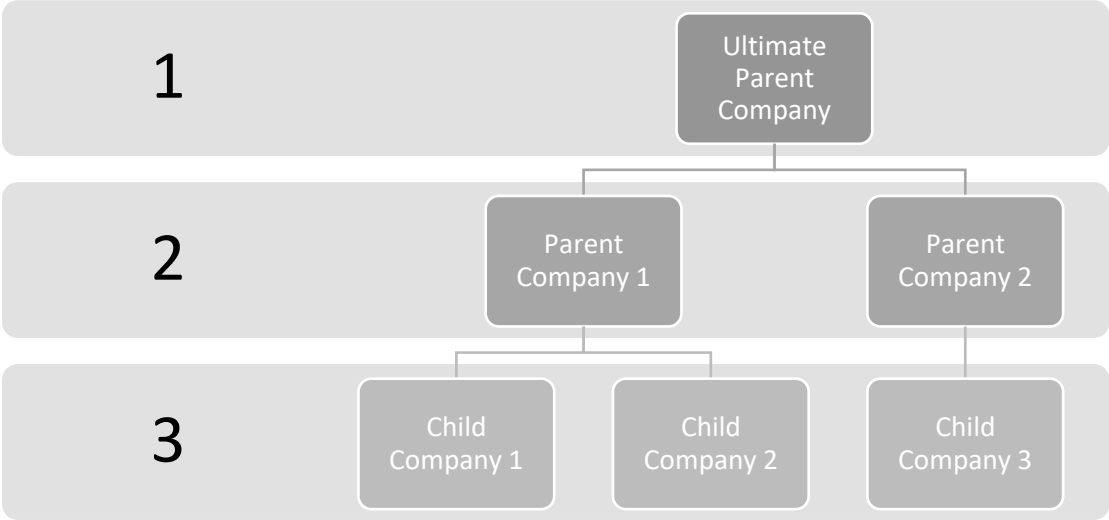


Figure 7: Hierarchy and Level of Companies in a Group Structure

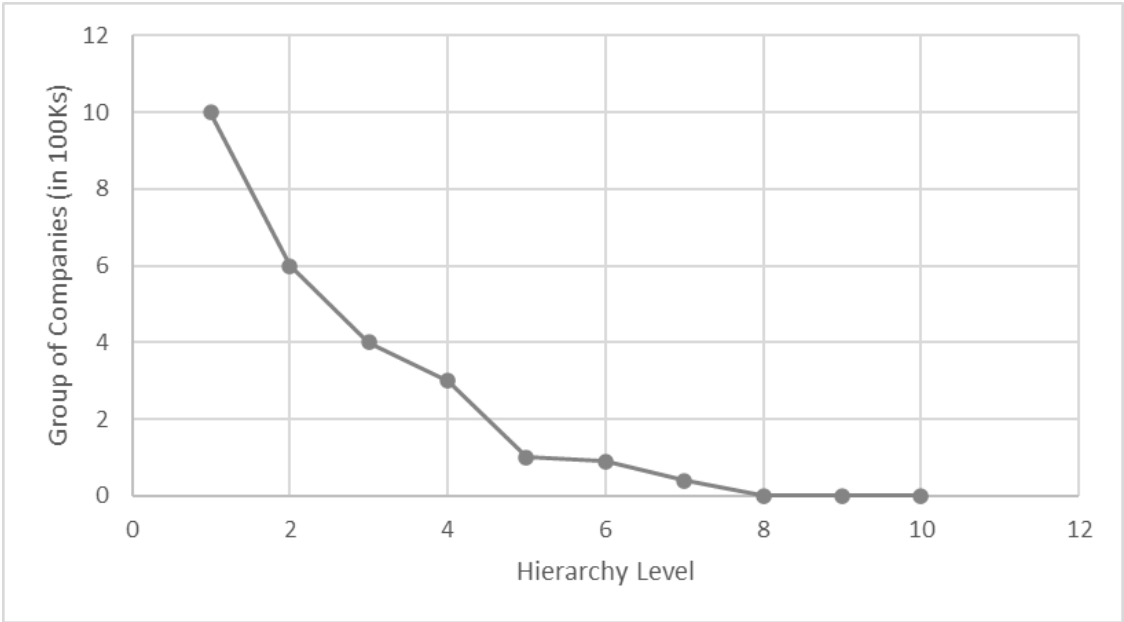


Figure 8: Line graph showing count of Companies (100Ks) for Hierarchy Levels

4. Most company relationships were found to be not explicitly defined in their ownership details. For example, a group of three companies viz. Bizibl Group Ltd, Bizibl Technology Ltd, Bizibl Finance Ltd, are owned by the two beneficiaries Mr Richard Jame Marks and Mr Jonathan Francis Bertram Hughes. For someone who understands ownership patterns or a lay man

looking at the following visual would easily understand that these 3 are part of the same group of companies but they are not related to each other as beneficiaries on their company details.

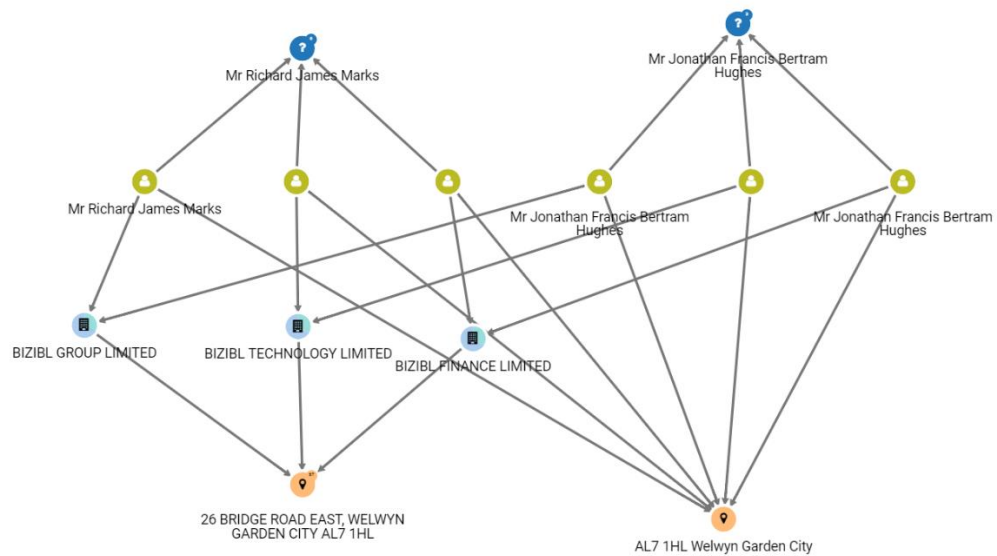


Figure 9: Investigating different group structures in Linkurious Platform

(Linkurious, 2023)

These kinds of relationships were derived from the Open Ownership by custom querying the data to replicate the relationships pattern.

5. Another challenge was to expand the relationships found and map that information to each child company. It was critical to have a macro view of the group of companies (grouping just 2 companies to 52K companies in one group) and also have a micro view where I can have the group information available for each of those group member child companies. To overcome this challenge, I had to take advantage programming in python pandas. Since the scale of the data is huge catering to big data, I had to research more on how it can be done efficiently without running out of computing power. The solution was to utilise some specific pandas' function which helped me perform the expansion (Efficient way to unnest (explode) multiple list columns in a pandas DataFrame, 2017).

6. With unfolding new insights and using transformation tools like python pandas and SQLite recursive querying it was mandatory to import and export data from python DataFrame to SQL back and forth. Again, achieving this with big data technologies was research in itself. Following best practices of SQLite pragma statements was critical and maintaining an orchestration pipeline which would cater to certain changes for effective performance became essential. A vivid set of queries were created to cater to the optimum use of the technologies.

Fundamental Challenges:

1. Defining a data structure and warehousing model was an essential so as to be agile and conscious about how one can better understand the flow of extract, transformation and loading of data. I had the liberty to build the data structure of tables, indexes, data types and warehouse as I needed but I wanted to keep it in line with what outcome was expected. I also wanted to eliminate as much manual steps as I could, this way the chance of manual errors would be diminished, and accuracy of the data could be benchmarked. I had to integrate some data from The Data City to retain some of their historical findings. So, the new insights had to be in line with the historical data.
2. Understanding how data behaves for the business was not that challenging but was mandatory so as to tailor my outcomes to the business expectations. This allowed me to understand how the data might be used on the platform and how it would behave when a user would see group companies instead of individual company records.
3. Orchestration pipeline design was one of the key elements as part of data engineering of the project. The idea was to create an orchestration pipeline which could be run over time with handling as much errors as possible. Also make it easy for anyone to read, understand and use it with a few clicks.
4. Data archiving was kind of an added layer of data warehousing which I integrated to allow TDC to maintain the code of the project and also maintain the results outcome databases which can be looked back to identify changes in ownerships if needed.

1.7.1. Availability of data

At the start of the project data availability was never a hurdle but it was essential to know that enough information is available to identify the group structures and create custom queries based on the available detailed information of companies. It was interesting to iterate the solution through the project lifecycle and see how recent changes to the BODS data format were handled seamlessly without changing the logic.

1.7.2. Visualization Data

Data visualization of the data as and when needed was required to better understand the group structure of companies. Although it was not essential to be done for each group structure. Hence, whenever needed I would utilise the Linkurious platform to visualise group patterns. It is out of scope of the project to focus on utilizing the data on TDC platform and create visually expandable group of companies because it has to be done with the help of TDC front end team. Understanding how their current structure works and then developing a prototype to understand how it can be scaled for all of the data.

Some insights from the company group structures.

Interestingly, the top 10 Ultimate Parent Companies from our data are the following. The biggest Ultimate Parent Company who has 2359 subsidiary companies is Harkers Associates Limited which is then solely owned by Mr Marc Anthony Feldman. I am amazed to see this distribution where the top 5th Ultimate Parent company CFS Secretaries has almost 3 times less sub-companies.

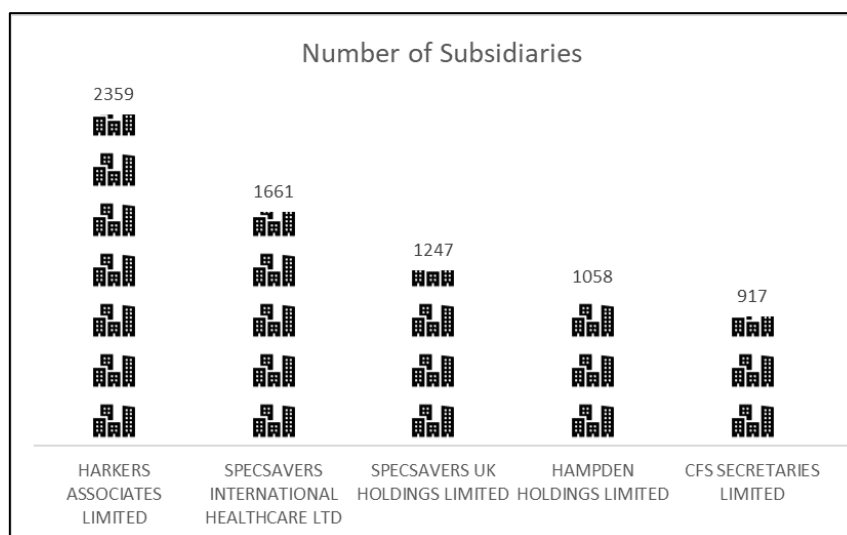


Figure 10: Top 5 Ultimate Parent Companies based on total count of sub companies.

The top 6-10 Ultimate parent companies have another story in themselves. The distribution here is equal but almost half the size of sub-companies the 5th top Ultimate Parent company and 5 times less sub-companies compared to Harkers Associate. For now, we can conclude that Harkers Associate Limited is the “Ultimate” Ultimate Parent Company leading the list of 269,738 Ultimate Parent companies which were identified as part of my exploration.

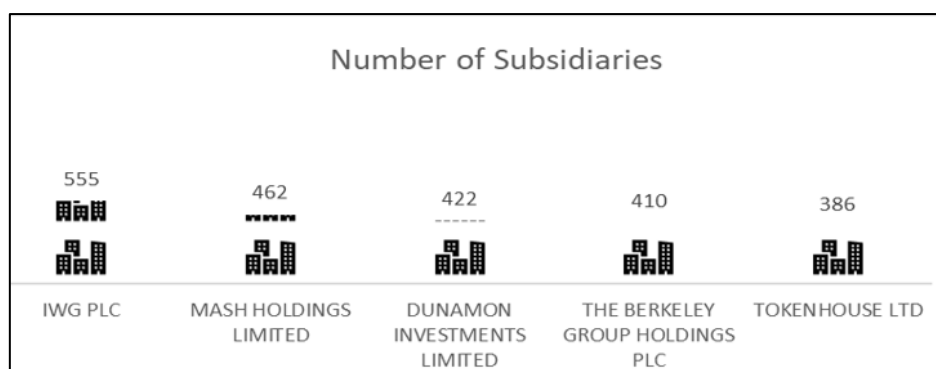


Figure 11: Top 6th-10th Ultimate Parent Companies based on total count of sub companies.

Another estimation that we do using custom relationships between companies and estimating their group structures shows is this insightful information. The top address that we see in London, WC2H 9JQ has more than 52K companies listed with over 53K beneficiaries. This is the most popular companies address in the Companies House registry office. Similarly, to our companies' groups structures our estimated group of companies have a skewed distribution.

The top 6th - 10th estimated group has 10 times less companies and beneficiaries. One thing to note that all top 10 estimated group of companies have an address in London, although the Open Ownership data is coming from Denmark, Slovakia, and the United Kingdom.

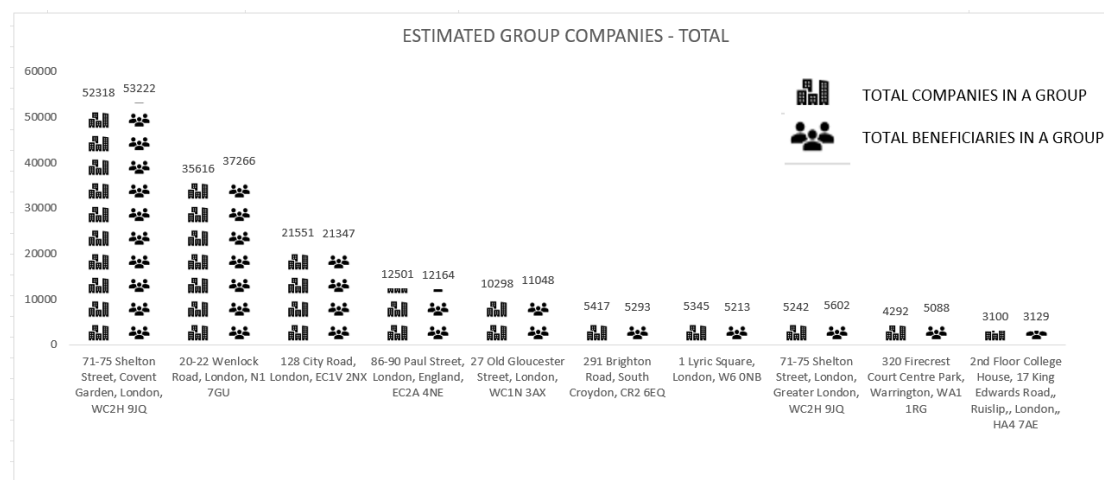


Figure 12: Top 10 Estimated Group Companies based on total count of companies and beneficiaries

1.7.3. Graph & Relational Database limitations

It is beneficial to visually understand that the data we are dealing with is structured to be a graph data which has nodes like person, entity, address, etc. and has branches like relationships, details, etc. This allowed me to understand how we can use the graph data but on a relational database. The end result that we wanted was to create a hierarchical table which essentially points to a different row entity as a parent of the current row entity. An example is shown below:

Ultimate Parent Company	Parent Company	Child Company
1	1	2
1	2	5
1	2	3

Figure 13: Company Group Structures as defined in our Data Tables

This kind of a hierarchical structure allows for the data granularity to be as detailed for viewing a child companies details but also have knowledge of their macro ultimate ownership. Here Company number 1,2,3,5 is representation of SIC codes and arrows are representation of their ownership relations.

1.7.4. Data Limitations

For now, the data is limited to Denmark, Slovakia, and the United Kingdom countries. This is because not all countries around the world have agencies like Companies House who make the beneficiary ownership data available to the public in the standard format. This limitation is a fundamental hurdle for all research in this area. The users who want this kind of relational data are inclined towards BODS formatting and are supporting heavily for countries to make their data standardized and available to the public (Strengthening new norms in beneficial ownership transparency in Asia and the Pacific region, 2022).

1.8. Development Methods

1.8.1. How to find Company Ownership Chains?

Using Open Ownership data, it is easier to find relationship using simple SQL queries when we are querying only one or a handful of companies. But when we want to scale the query to millions of companies and their sub-companies, the query needs to improve to handle the volume, errors and generalise the query for different company detail patterns.

The way I found companies group structures was:

1. Find company details by joining data from entity_statement table and entity_identifiers table.

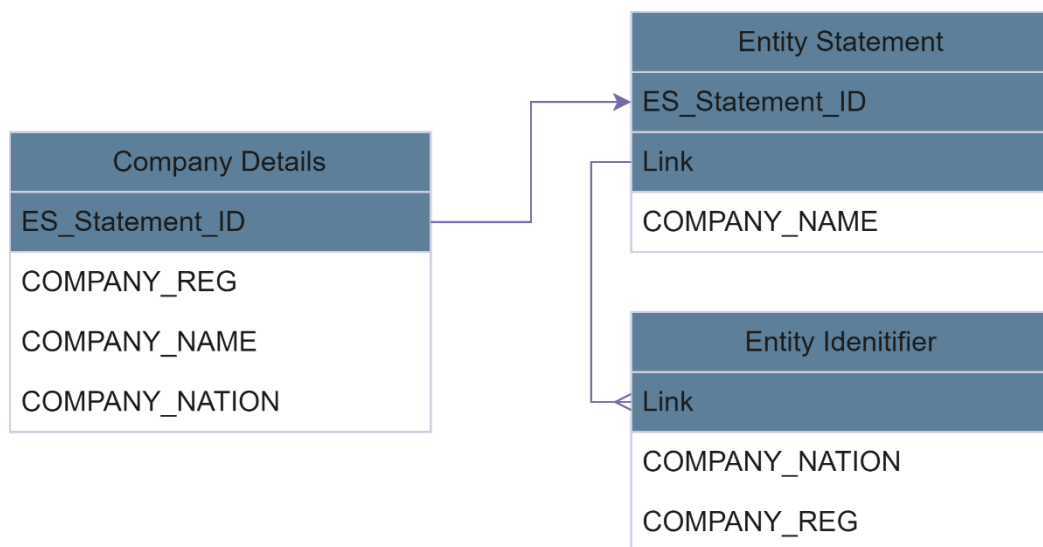


Figure 14: Stage 1 finding of Company Chains

2. Find relationships between each company and their listed beneficiaries (person and entity) by joining the staging table created in step 1 with ownership_statement and ownership_interests table. The table found here is enough to find relations within company

group chains because this table is similar to having hierarchical information linking to each other row in the same table based on the company's registration number.

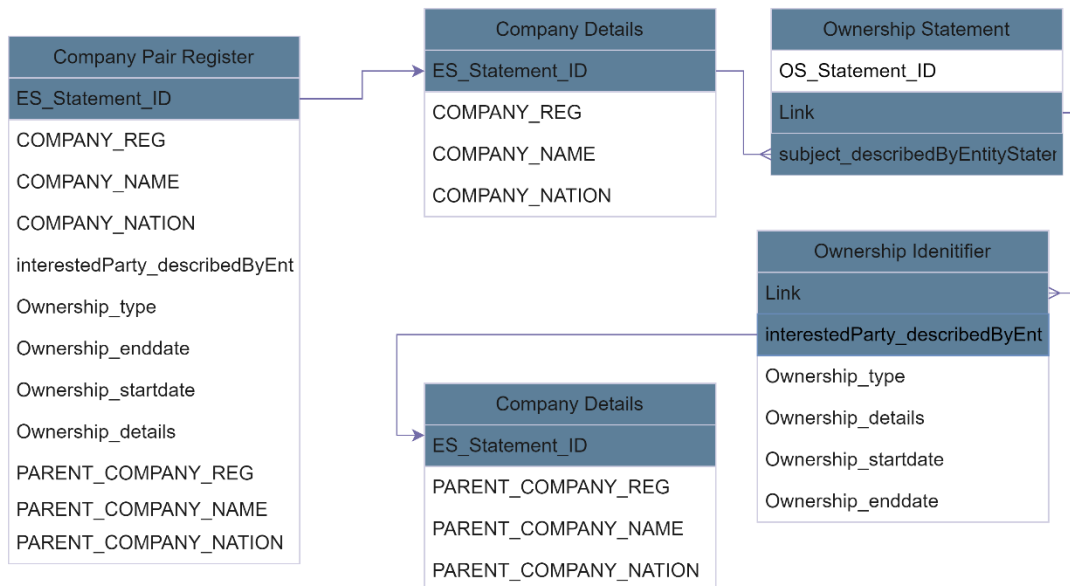


Figure 15: Stage 2 of finding Company Chains

Initially the idea was to go bottom up and find ultimate parent for each company in the table found in step 2. But SQL querying this way would take ages to complete because it would never know which company is at which level in the hierarchy and when it needs to stop recursively finding more pairs (stop scanning the same table over and over again).

3. Hence, the 3rd step is another staging layer to only identify the companies which are top in their group chains also known as Ultimate Parent Companies. These companies are the entities which are not present in the Child Company Column in our Company Pair register and are only present in the Parent Company Column.

This step helps us in multiple processes in the project ahead. For example, validating the count of group structures found, using this table in the use case queries for faster results, etc.

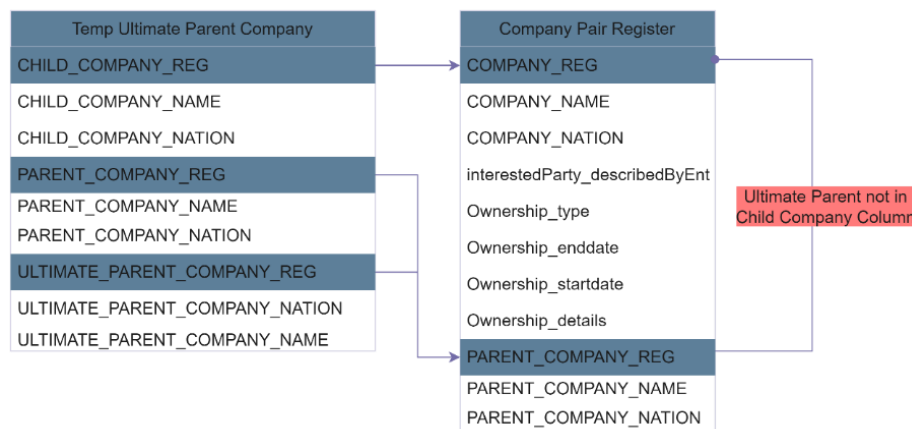


Figure 16: Stage 3 Finding Ultimate Parent Companies

4. The final step is to use the Ultimate Parent Staging table as the base of recursive query and find all the sub-companies of each Ultimate Parent and each sub-company then found. This could be imagined by understanding a function which basically when given a company number will find all the Child Companies it has referring itself in the Parent Company Column in the from the Company Pair Table. Since, this is a recursive query I had to limit it to a certain layer of hierarchy below which it would not get any data but would infinitely be looped if there would have been a circular dependency of company relations.

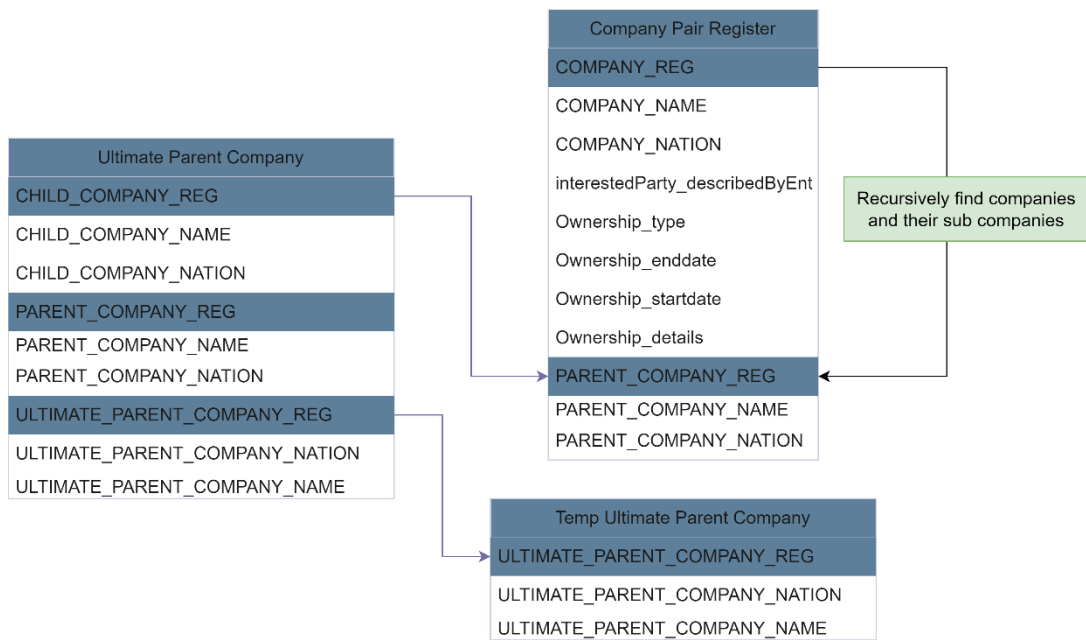


Figure 17: Final Stage Finding all sub companies using SQL recursion

1.8.2. Why recursion?

Several different algorithms, approaches and functions were discussed.

1. Bottom-Up Approach: This was the first thought in my mind because of the nature of the data where we had companies and had found their immediate owners and beneficiaries. But there were a few problems in this approach as the algorithm was not aware if it is really at the bottom of the group structure or somewhere in between. In other words, each company pair would have more beneficiaries up in the hierarchy and more sub-companies in the hierarchy below. This approach hence became complex because querying in both directions until the top and bottom would be difficult.

2. Bottom-Up recursion: Another approach I thought of was to first find all child companies who do not have any further sub-companies then use a recursive query which would run upwards

trying to find a parent company and chain of companies further to the ultimate parent company. This approach was simpler compared to the approach above but had some limitations to it. The biggest limitations were that for each company it would look up for only one parent and ultimate parent. This was difficult as a company might have more than 1 beneficiary and I wanted to record each of these company paths to all the ultimate parents.

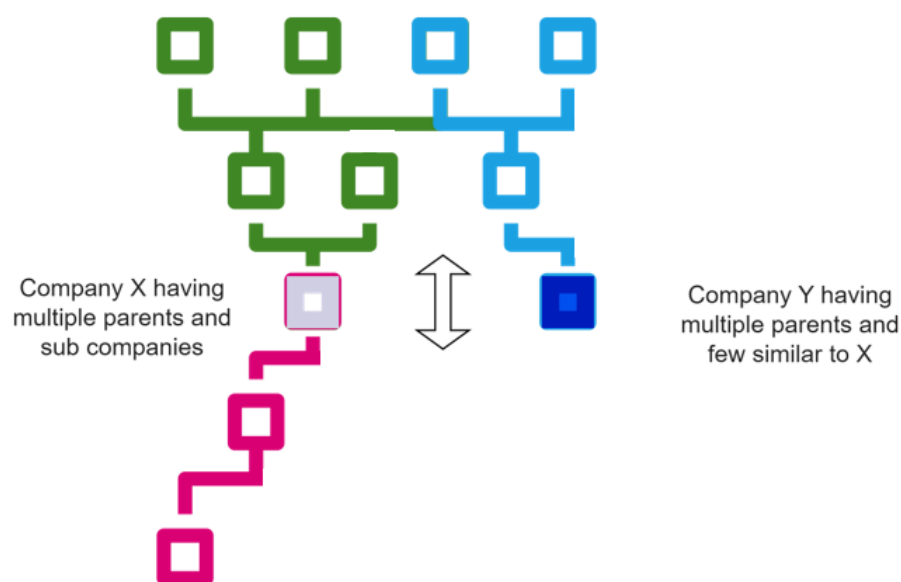


Figure 18: Visualising Companies in group chains with companies above and below in the hierarchy

3. Top-Down Recursion: This method was successful for our findings as we started with all the ultimate parent companies and then found all the sub-company and chains for each of these ultimate parents. This was we might see a child company being listed under one or more immediate parent and ultimate parent company and this is what was expected for the output.

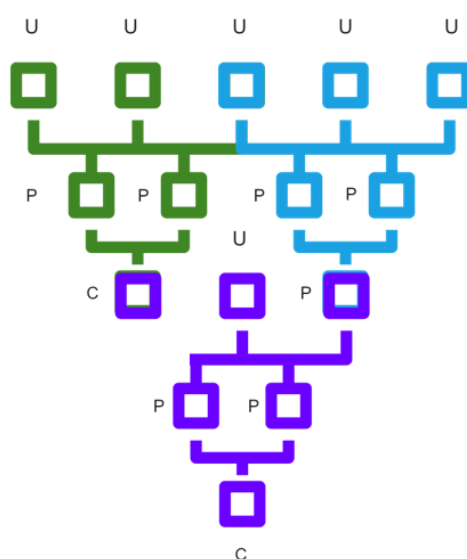


Figure 19: Companies in group chains starting from U - Ultimate Parent through P - Parent Companies and to C - Child Companies

1.8.3. What other Group Structures to find?

As part of the phase 2 of the project we decided to find group structures which could not be defined using beneficiary ownerships as these chains are not explicitly accounted in books. For this I used Linkurious visualisations to find companies which could be grouped in one based on the people that own the company and the addresses they are listed on. This was fairly simple to query and gave more insights into the company ownership data. This is a way to estimate the company group structures which may be part of the same group of companies as they are owned by the same pair of person owners, listed at the same company and the owners also have the same address. It was interesting to see the number of companies that were grouped by this way was more than the number of companies in company chains.

2. Project Management

The project in all was managed in an agile manner having weekly sprints to understand the progress, findings, doubts, and gain feedback to define new goals for the upcoming sprints.

2.1. Overview

To give an overview of what steps were recorded, monitored, and achieved throughout the project life cycle I had created a dashboard as follows.



The image shows a screenshot of a Kanban dashboard titled "Cleaning Data and Data Exploration". The dashboard is organized into columns: "Task", "Assigned", "Task progress", and "Timeline". Each task row includes a checkbox, a task description, an assigned person icon, a progress status, and a timeline range.

Task	Assigned	Task progress	Timeline
Format data into flat file	BS	Done	-
Identify the parent for every company	BS	Done	Jun 1 - 23
Identify the ultimate parent for company (with ...	BS	Done	Jun 1 - 24
Understand OpenOwnership data	BS	Done	May 4 - 24
Read article #1	BS	Done	-
Examples of problem companies	BS	Done	-
Upload Shareholders data	BS	Done	-
Methodology review/Code Review	BS	Working on it	-
Validating UAT parent-child- Ultimate parent d...	BS	Done	-
Standardize and process the ETL model	BS	Done	Jun 12 - Aug 11
Creating a Github Repo for the model	BS	Done	-
Phase 2 - Extracting Cluster Pairs for Companie...	BS	Done	Jul 17 - 28
Validating the Phase 2 Company groups extrac...	BS	Done	Jul 31 - Aug 7
Updating the Github model for better readabilit...	BS	Working on it	Jul 24 - Aug 7
Eliminating the manual steps	BS	Done	Aug 14 - 21
Archive and Create copy of the final table	BS	Done	Aug 14 - 21

Figure 20: List of tasks listed on Kanban Dashboard for tracking development.

2.2. Time management and sprints

It was critically important to manage time and perform certain tasks to learn through the progress of the project and act upon activities and tasks as per the learnings. Weekly sprint meeting with The Data City helped me understand their expectations and how well the progress done over the week was. At the start of the project good amount of time was invested in searching solutions to transforming the JSON formatted Open Ownership Data later realising that it was available in other feasible formats. This helped us take a step back and think thoroughly on whether other such tools should be researched and how this research-based pause to the development would impact the timeline. But eventually with the tools I had I was able to resume with the development and complete the findings in time. After the phase 1 findings we had enough information to validate the data and group structures found. Then to realise that using Linkurious we can find more structures which are not identified through beneficiary details of the company. So as part of the phase 2 with the findings of phase 1 I developed custom query to replicate relationships found on Linkurious.

2.3. Kanban

We had created a dashboard on which tasks and future sprint goals were defined. This way I was able to prioritize essential, urgent, and pending tasks. This also created a sense of discipline for me in terms of developing and documenting the findings. Also, with sprints reporting every week I had to explain the key findings and hurdles to get feedback and help as and when needed.

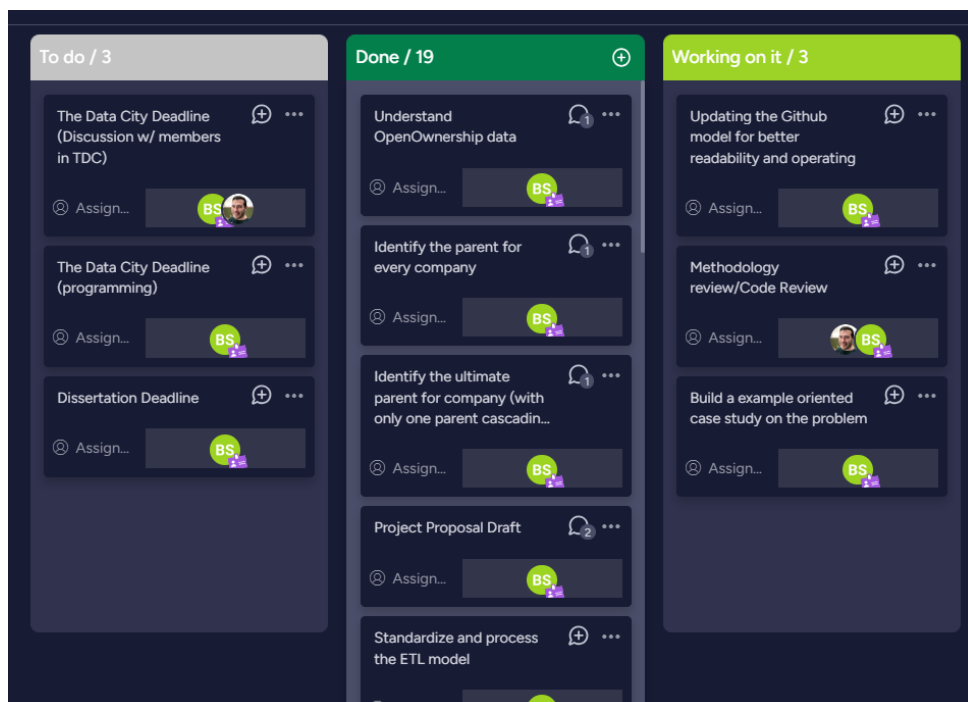


Figure 21: Kanban Chart of Project Tasks

2.4. Integration with The Data City

Integration with the team at TDC was one of the easiest and smoothest process. As I could get in touch with them through team's chat and also learn from them using their best practices and learnings. I also got the opportunity to visit them at their office and work with them actively on the tasks for the week. This allowed me to understand the project problem more and also the overall impact on the business with the findings. As part of the future scope, we still have to integrate the developed solution into their platform. We had also scheduled meeting to validate the methodology of the project solution, running the orchestration, validating the data, and clearing any doubts they might have before I hand over the project. Basically, a knowledge transfer session was conducted for explaining how to use the solution.

3. Architecture, Design, and Implementation

3.1. Overview

Architecture of the whole data solution is built in stages. Where the data extracted from the Open Ownership register is to be stored in a separate database, the medieval stages are stored in a different database and the master tables with group structures are copied into a different database for utilisation on the platform.

Design of the databases and table is represented as follows.

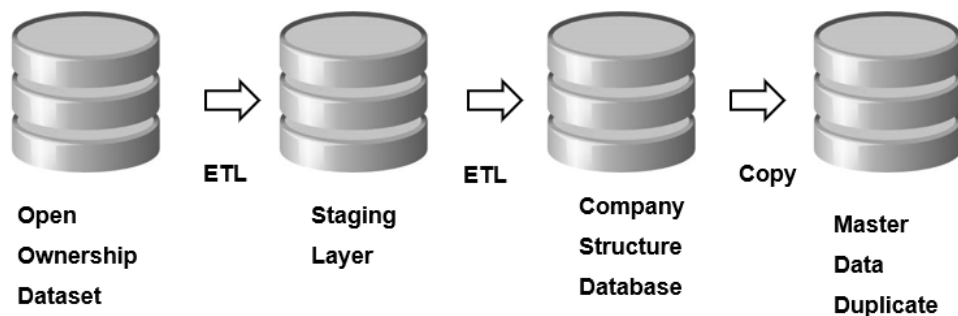


Figure 22: Extract Transform and Load Process of Project

Implementation of the ETL transformation, storing of the staging data, copying of the master data is done by using python notebooks in conjunction with SQLite, pandas, and other modules.

3.2. Data Extraction

Extracting the relevant information from the Open Ownership Dataset, I tried to think through on what information about the relationships and entities would be beneficial in the later stages of the project but had to go back and iterate over the staging layer to get more information as

and when needed. Doing the project in stages helped me break down the extraction into pieces and focus on getting accuracy of the data. The data from the Open Ownership dataset had to be revised in terms of their data types, indexes, etc. This was essential for the extraction to be efficient, meaningful, and usable later in the stages.

3.3. Choosing data source types and granularity

The data type and granularity of the data was one important factor when looking up relationships and validating them. SQLite had this minor limitation for comparing text-based data types at this huge data scale. It would struggle to join tables based on columns which had text data type. Hence a solution to this was to redefine the Open Ownership data tables with data type text with no case collation. This would allow SQLite to ignore the case sensitivity of the data and join data tables faster.

3.4. Data Storing

Data storing was always thought through to be stored in form of tables in a database with efficient indexing. This was understood to be part of the outcome as it would make it easier to integrate the data with TDC platform's existing infrastructure. Hence, all of the data is stored in SQL tables inside a database. An additional advantage is that this can be compressed for saving storage capacity and efficiently sharing within systems.

3.5. Data Indexing

Indexing makes columns faster to query by creating pointers to where data is stored within a database. An index is a structure that holds the field the index is sorting and a pointer from each record to their corresponding record in the original table where the data is actually stored. Indexes are used in things like a contact list where the data may be physically stored in the order you add people's contact information, but it is easier to find people when listed out in alphabetical order. There are two types of databases indexes Clustered and Non-clustered.

3.5.1. Index and Query Performance

In the project, since we were searching for information from tables having more than 12M rows it was essential to create indexes for efficient and fast querying of the data.

I had to create 29 different indexes for around 10 different tables for efficient querying. My initial approach to select columns to index was:

1. Index all columns frequently used for querying and joining tables.

2. Get an explanation of the query plan which took long time to execute using EXPLAIN QUERY PLAN function of SQLite.

3. Use the information to better understand where a compound index might be helpful.

Indexing drastically improved the speed and performance efficiency of the SQL queries.

One interesting event was to see the query use up over 100/120GB of RAM memory from our virtual desktop when I was building a complex query without indexes.

Indexing also helped me to get to the root cause of complex query which took long time.

1. Here, I already had indexing in place for all the relevant columns.

2. The query was taking longer because the columns were case sensitive, and I was querying to join on this specific name column.

3. Then I had to remove the case sensitivity of the name columns for faster querying

Check appendix to see which columns were indexed.

3.6. Data transformation layers

Data transformation layers as shown earlier helped the project visualise the progress of the data transformation. Also, helped me pause and resume the data findings from the staging tables. After completing phase 1 of group structure findings, I was able to re-use the staging tables in the phase 2. This made it possible to complete phase 2 in less time compared to phase 1 because a lot of staging defined tables were re-purposed.

Data transformation layers built as part of phase 1:

1. The information for each company which was relevant to finding company chains and structures was extracted from different entity tables in the Open Ownership dataset. Each company required company name, company registration number, company nation and entity id from the entity tables. At first the company registration number was to be used as a unique identifier for companies, but it was found to have duplicates because the data is from three different jurisdictions using the same length of registration numbers hence entity id uniquely identified by Open Ownership was used.

2. This information for each company was then used to merge it with ownership tables to identify companies and their immediate beneficiaries and owners. This staging layer allowed to form the base of our investigation further. For each company we gathered the information about its ownership type, ownership start date and end date, owner name, owner identifier (defined by Open Ownership), and the data was filtered to companies having only entity as owners and not person owners as part of phase 1.

3. The third stage was to create a list of all ultimate parent companies which were found as parent companies and not as a child company in the stage 2 above. This staging table became the start of our recursive query.
4. Final stage was to use the recursion query and find company chains for each of the ultimate parent companies and their sub companies listed in the stage 2 layer. This final table was all the information that we needed as per the expected output for the project.
5. After the final stage we had to merge the data with the existing company group chain data available and create a master table which could be directly used on the platform.

Data transformation layers built as part of phase 2:

1. As part of phase 2 first we used the same stage 1 table already built for phase 1.
2. Next we found companies who were owned by person entities and were not grouped using phase 1. For each company we gathered the information about its ownership type, ownership start date and end date, owner name, owner identifier (defined by Open Ownership), and the data was filtered to companies having only person as owners and not entity owners as part of phase 2.
3. This stage was to group all the companies who had
 - a. The same person owners.
 - b. The same company listed address.
 - c. All the owners had the same person address.

This grouped all the companies and gave us a unique group identification for each group and listed all the companies and beneficiaries in one column.

4. Final stage was to expand this grouped information to list all the companies and give information about count of companies which are estimated to be the part of the same group of companies and count of beneficiaries of the group. This was trickiest query to be built in SQL as expanding row values was difficult with the built-in functions and hence, we used the power of python computing using pandas explode function.
5. After the final stage, we just had to copy the expanded table to SQLite for easy access later.

3.7. Merging relevant data and ETL orchestration design

Before the project TDC were sourcing this information from a different source and already had some findings which were authentic and validated and already in use on the platform. The idea was to use Open Ownership data and scale the findings. After completion of phase 1 of data where company group structures were available in a similar format as to the existing data. It

was essential to merge the data and keep relevant information from both the sources. The ETL orchestration was designed considering this merging step and was smoothly taken care of at the time.

3.8. Data Output and Further Integration

The output of the data group structures is designed to be integrated with TDC's platform. The idea is.

1. For each company to have information about their immediate parent company and ultimate parent company.
2. When the company is not owned by another company, and it is part of a group of companies like our BIZIBL group example it should have information about which companies are part of the group and how many beneficiaries are owning the group.

For example, from the platform:

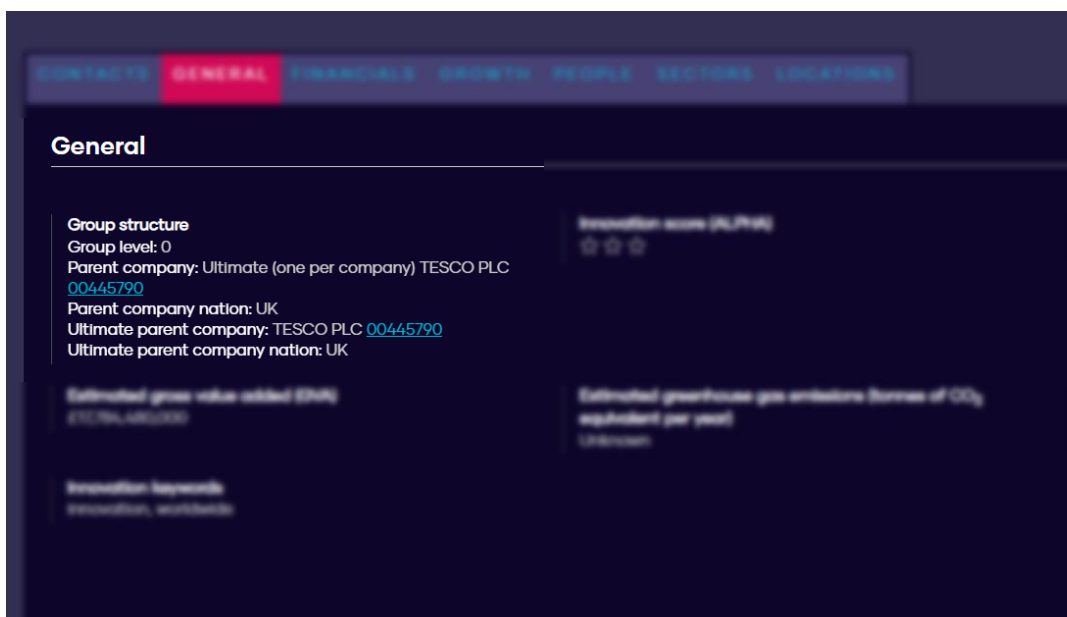


Figure 23: The Data City Product - Company Group Details

(The Data City, 2023)

3.9. Language

The computing language used for all the transformation is SQLite and the programming language used to orchestrate the data flow is python. A combination of SQL and python is used to get all the advantages of transformation that are possible with both languages.

3.9.1. SQL

A relational database's structured query language (SQL) is a programming language used to store and process data. In a relational database, data is stored in tabular form, with rows and columns denoting various data qualities and the relationships between the values of those attributes. To store, update, remove, search for, and retrieve data from the database, utilise SQL statements. SQL can also be used to optimise and maintain database performance. A well-liked query language that is commonly used in many kinds of applications is structured query language (SQL). SQL is a popular programming language among data analysts and developers because it interfaces well with other programming languages. For instance, users can use the Java programming language to incorporate SQL queries in order to create high-performance data processing applications using popular SQL database systems like Oracle or MS SQL Server.

SQL statements, or SQL queries, are valid instructions that relational database management systems understand. Software developers build SQL statements by using different SQL language elements. SQL language elements are components such as identifiers, variables, and search conditions that form a correct SQL statement.

3.9.2. Python

A high-level, all-purpose programming language is Python. Code readability is prioritised in its design philosophy, which makes heavy use of indentation. Python uses garbage collection and has dynamic typing. It supports a variety of programming paradigms, including procedural, object-oriented, and functional programming as well as structured programming (especially this). Due to its extensive standard library, it is frequently referred to as a "batteries included" language.

Python Pandas:

Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with "relational" or "labelled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open-source data analysis/manipulation tool available in any language. The two primary data structures of pandas, Series (1-dimensional) and DataFrame (2-dimensional), handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering. For our project we used DataFrame for most use cases.

3.10. Database Software

A self-contained, serverless, zero-configuration, transactional SQL database engine is implemented by SQLite, an in-process library. Since SQLite's source code is in the public domain, it may be used for any objective, whether public or private, which is perfect for our project. An embedded SQL database engine is SQLite. SQLite lacks a dedicated server process, unlike the majority of other SQL databases. SQLite directly reads and writes to common disk files. One disk file can store an entire SQL database, including all of its tables, indices, triggers, and views.

3.11. Algorithms and Functions

Recursive Querying in SQLite: The SQL feature known as the CTE (common table expression), commonly referred to as the WITH clause, returns a temporary data set that can be used by another query. It is not kept anywhere because it is a temporary result, but it can still be referred to just like any other table.

For our project we used CTE to identify relationships between different rows within the same table. The idea was to either go bottom up or to go top down to find relationships and chain of companies. The top-down approach was beneficial for the project as this way we could have.

1. The distinct ultimate parents in the relationship tables.
2. Use these ultimate parents to identify their chain of companies.

3.12. Setup & Installation

For the project one has to setup and install python on their machine to be able to run the python notebook and also install relevant modules needed for the orchestration. One can also install dB browser to view the extracted data tables.

Through the orchestration process I have mentioned steps to

1. Create indexes
2. Create tables
3. Attach databases
4. Download the Open Ownership dataset and extract it.
5. Recreate a few tables from Open Ownership dataset to adhere to later transformations and joins.
6. At the end of the orchestration I have added steps to save changes and close the databases.

4. Use Case of Analysed Data on RTIC Lists

I had to use the group structures on a RTIC list and understand user acceptance testing,

4.1. Use case problem.

Understanding the use case problem is key. For example, I tried to look at one of the RTIC list called Streaming Economy on The Data City's platform. This list of RTIC has companies who have similar business in the streaming economy. I am no expert to further explain the fundamentals of it, for the project perspective this is just a subset of all the 5 million companies that The Data City has information about. This Stream Economy RTIC has around 604 companies.

4.2. Confusion for end user

The user starts going through all of these companies, filters few of them based on their expertise using the companies details like financials, sector, growth, location, SIC codes, etc. The user ends up with a niche list of companies which they would like to contact for business. The users try to contact all of these companies only to realise the 20% of the company contacted redirected them to either their subsidiaries or parent companies as they cannot provide more information on the business query the user had. This is a classic example where the user comes back to The Data City to ask if this redirection effort loss can be reduced with getting more insights on the company's granularity, hierarchical level, and ultimate parent information. A user looking at all of this would want to know more about which companies in the list can be grouped based on their beneficiaries even before starting to analyse the results further or contacting all these companies for business. This is the basic confusion for end user summary of our problem.

4.3. Solution provided.

The Solution that we have provided for the end user is:

1. Join the company group structures we have identified with our solution with the list of companies in the RTIC.
2. Link the list of companies from the RTIC to find if an immediate parent or an ultimate parent is available in the list.
3. Check if the company's website URL matches with its parent company's URL.
4. After joining and checking the data we allow for the user to filter the data based on their requirement with a filter feature.

5. When they want child companies, they can filter the information to exclude all parent/ultimate parents from the list.

6. When they want ultimate parent or parent companies, they can filter the list to exclude the child companies from the list.

Following is an extract of our Streaming Economy RTIC Use case.

The company structure data looked up from our solution are the first two columns showing the immediate parent and ultimate parent company numbers. The flags that we have calculated using a custom SQL query are show in the next four columns what they denote is as follows.

Flag name	Definition
I am Parent	When the company is in the list of parent companies in our group structure this flag is set to 1.
I have Parent	When the company has either their parent or ultimate parent in the list then this flag is set to 2, 1
My Parent URL	Looks up the URL of the Ultimate Parent or Parent Company
Exclude Or Not	When the company has a different URL to the parent or ultimate parent then this flag is set to 0 either it copies the value of "I have Parent" flag.

Table 1: Flags and Definitions in Use Case Problem

PARENT_COMPANY_REG	ULTIMATE_PARENT_COMPANY_REG	I_am_parent	I_have_parent	My_Parent_URL	Exclude_Or_Not
		0	0	dandelion-burdock.com	0
		0	0	zcomax.co.uk	0
		0	0	surroundvision.co.uk	0
		0	0	dlt.com	0
2275557	#0006442	1	2		2
2226228	#0006442	0	2		2
2226228	2226228	0	1		1
2270255	2270255	0	1	technicolor.com	1
#0036017	#0006442	1	0	technicolor.com	0
#0036017	#0006442	0	0	technicolor.com	0
303309	#0006442	0	2	technicolor.com	2
303309	303309	0	1	technicolor.com	1
		1	0	thebroadcastbridge.com	0
8815019	8815019	0	1	thebroadcastbridge.com	1
#0088749	#0088749	0	0	corp.kaltura.com	0
		0	0	isize.co	0
		0	0	course5i.com	0
#0131338	#0131338	0	0	exasol.com	0
		0	0	quix.io	0
6606840	#0005784	0	2		2

Figure 24: Data and Flags Updated of Use Case Problem

4.4. Time saving

The user now looking at this data has additional information about the company's immediate parent company, ultimate parent company and flags that we have created. This allows the user to philtre through irrelevant companies based on the granularity they want. Help them focus on

the companies that are relevant to their business need. Helps them direct their business to the companies that can impact their business. When the user uses this new data, they allowed themselves to understand the company group structures and contact only the relevant companies.

4.5. Analysis Potential

The potential of the analysis is to be able to use this company group structures and estimate company financials, employee count, growth, and other properties this will enhance the user experience in terms of understanding company group structures and individual company performances. This in turn will allow the user to get more insights on the company's businesses and make effective decisions when trying to contact these companies for their own your winner looking better business.

Other potential uses of the data would be to use the company group structures and move out of the RTIC list and find the ultimate parent companies. This way the user journey jumps from their recent RTIC search and takes a new pivot to learning more about the ultimate parent company and their business. This allows users to see the bigger image and potentially increase the scope of their business purposes. This has immense potential in terms of marketing, networking, generating leads, expanding their own business vertically, horizontally, and internationally.

5. GitHub Version control

For version control and a standard development approach we used GitHub to store the queries and codes created.

5.1. GitHub

A web-based platform for collaborative software development and version control is called GitHub. For their code projects, it enables both individuals and teams to set up repositories, allowing for effective change tracking over time. Through pull requests, developers may quickly submit, review, and integrate code, promoting teamwork. A streamlined development approach is supported by GitHub's seamless environment for issue tracking, document sharing, and continuous integration. It is a cornerstone of contemporary programming thanks to its user-friendly interface, Git integration, and large community, which encourages open-source contributions and makes code management possible everywhere.

5.2. How version control helped in the project?

Version control was the need of the project from the first few weeks. As I started creating queries and extraction procedures to find relations between companies, I wanted the ability to store the queries and improve upon the latest findings. Also, I wanted to store the findings for each query and notes on why a certain change was made and how it was necessary in the project lifecycle.

Using GitHub integrated inside Visual Studio Code helped me.

1. Track the changes made
2. Stage the changes
3. Sync them with the repository.
4. Make different branches of this code to develop new ideas separately.

Other advantages of using GitHub for version control is that after a few iterations of multiple uses of the code if need comes following updates to the code can be integrated.

1. Improvements or changes to the code.
2. New functions and new ways of grouping can be updated.
3. If someone uses the code to find out other new use cases, they can also be integrated with a request made to the authors.

Additionally, when the team at The Data City wants to look back at certain code and understand why certain logic was built and what was the purpose of it then they can review the change history in the GitHub repository to better understand and get clarity.

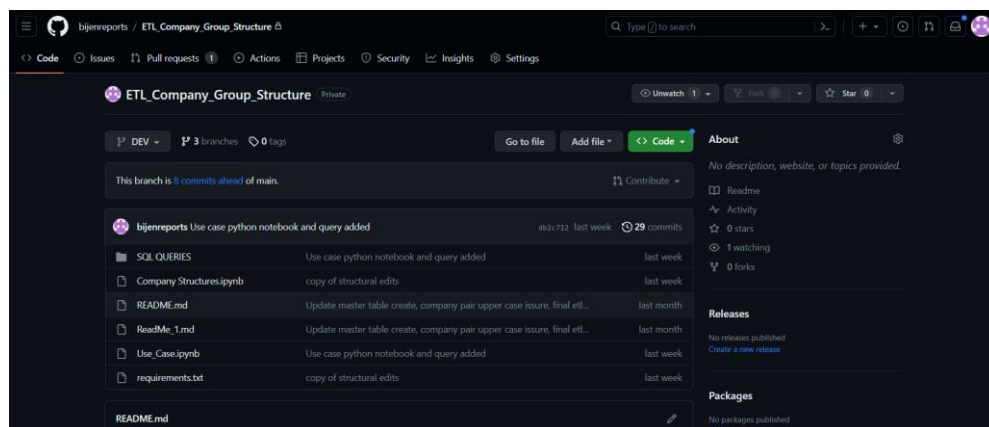


Figure 25: Github Repository of the Project

6. Future Scope

Future scope of the project is to visualise this group structures on the existing TDC's product and enhance user experience. The product team is now understanding the use of the data and how it can be integrated with their existing infrastructure. The output of the solution provided is made keeping in mind the final integration with the product.

6.1. Potential

- The potential use of the company structures information is to visually help the users to create new journeys when they are using The Data City product.
- Allow the users to estimate the company's financials, employee count, growth, etc. Understand how future acquisitions are impacting the companies' details and who is benefiting from the business.
- Create new marketing strategies to cater to this new finding. Potentially increase their scope of business vertically, horizontally, and internationally using the new group structure findings.
- Making strong connections with their existing customers based on the company group structure and bring in more business for their company and customers.
- For investors looking at this data, this will allow them to focus their investment strategies to the granularity that they want and benefit from the existing companies they are invested in.
- For recruiting purposes, the candidate can use this information and make more informed applications to not only the company but all other companies in that group structure.
- The potential use of this data is endless depending upon the users.

6.2. Project Success and Motivation

I am proud that the learnings from this project were one of the many motivations of The Data City's global product which is aimed at companies outside of UK. Although the product is still in beta version, I am sharing the suggested user interface for the product.

This is out of scope of the project, but it is exciting for me to acknowledge that the discussions and learnings from our project became one of the bases for their other products.

As you can see that in the new global product, we are already allowing the user to see the subsidiary companies which are available in the data.

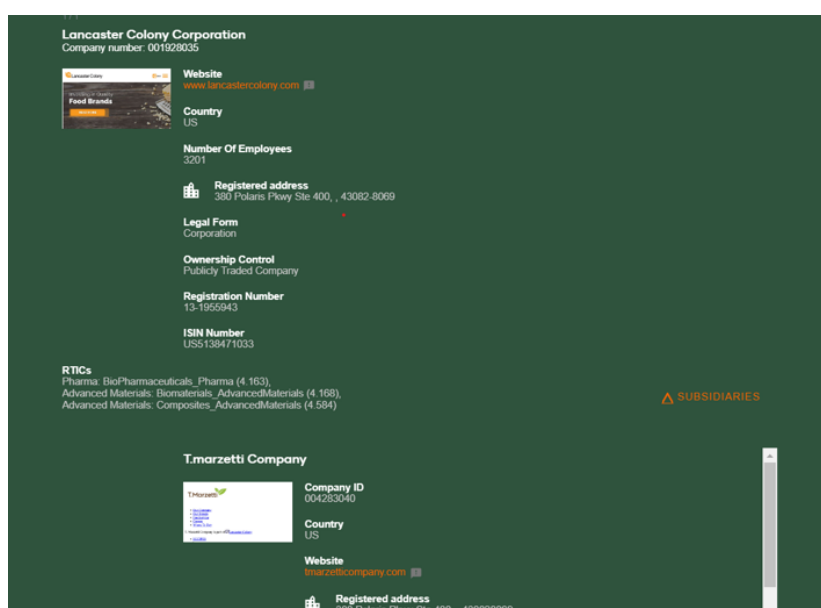


Figure 26: The Data City Global Product (Beta)

6.3. Limitations

The company group structures that we have found have some limitations for now.

1. The data is restricted to the 3 jurisdictions viz. UK, Denmark, Slovakia which limits our area of research and data to only 3 countries.
2. There could be more investigation done on identifying company group structures which are unique in themselves and not identified until now.
3. Relationship between a company's people beneficiaries and people beneficiaries of other companies and their ownership can be investigated to make better group structures.
4. The people's beneficiary data lacks information to uniquely identify a single user and can have duplicate names for the same person in different name structures. This creates confusion and limits are findings to define accurate group structures.

5. Research can be done on identifying companies details in the jurisdictions where the company beneficiary's ownership data is not available in BODS format. This will increase the scope of the data immensely and the learnings can be replicated for other such areas.

6. Well-structured data for now is beneficial but we may see lots of duplications and validation errors when we scale this project to companies all over the world. As I anticipate unstructured data and complex data definitions from different jurisdictions.

7. Conclusion

In conclusion the project is found to be of great importance and adds value to The Data City's data, knowledge, and learnings. The successful findings of the company group structure built more confidence in the team on utilising company group structures and fundamentally enhancing the user interface and experience.

The company group structure thus found will allow their existing users for an insightful update to their current user journey with The Data City. For new users it will become a base of what The Data City's product can do for their business research, marketing strategies, investments, recruiting, etc.

Starting out with the idea to use Open Ownership data from Open Corporates for finding company group structures, through identifying new ways to group companies based on their beneficiaries and to merging all this data to the existing group structures available. Overall, it has been a great learning experience for me and The Data City. The learnings and findings that we have developed will help everyone involved in their future projects.

The project gave me an inspiring experience working with The Data City on this project. Understanding TDC's way of working. Analysing their customer's requirements and researching relevant sources and tools which might help us through the project.

Developing a robust ETL transformation process which can be re used, improved and re purposed. Getting feedback on the findings from the user perspective and improving the results accordingly. Becoming one of the motivation factors for their new global product.

References

- Beneficial Ownership Data Analysis Tool*. (2023, 07 26). Retrieved from Open Ownership: <https://bods-data.openownership.org/source/register/>
- Companies House. (2023, 08 25). *Companies House*. Retrieved from Companies House: <https://find-and-update.company-information.service.gov.uk/company/00445790/officers>
- DB Browser for SQLite*. (2023, 05 04). Retrieved from [sqlitebrowser](https://sqlitebrowser.org/): <https://sqlitebrowser.org/>
- Economic Crime and Corporate Transparency Bill 2022: Factsheets*. (2023, June 20). Retrieved from Government UK: [https://www.gov.uk/government/publications/economic-crime-and-corporate-transparency-bill-2022-factsheets/factsheet-beneficial-ownership#:~:text=Beneficial%20ownership%20means%20those%20who,the%20legal%20owner\(s\).](https://www.gov.uk/government/publications/economic-crime-and-corporate-transparency-bill-2022-factsheets/factsheet-beneficial-ownership#:~:text=Beneficial%20ownership%20means%20those%20who,the%20legal%20owner(s).)
- Efficient way to unnest (explode) multiple list columns in a pandas DataFrame*. (2017, 08 23). Retrieved from StackOverflow: <https://stackoverflow.com/questions/45846765/efficient-way-to-unnest-explode-multiple-list-columns-in-a-pandas-dataframe>
- Linkurious. (2023, 08 20). *Linkurious*. Retrieved from Linkurious: <https://linkurious.com/>
- Open Ownership Register*. (2023). Retrieved from Open Ownership: <https://register.openownership.org/download#what-is-it>
- Optimizing a recursive CTE*. (2021, 09 28). Retrieved from Stack exchange: <https://dba.stackexchange.com/questions/300260/optimizing-a-recursive-cte-or-replacing-it-with-a-temporary-table>
- Phiresky's Blog*. (2020, 06 26). Retrieved from github: <https://phiresky.github.io/blog/2020/sqlite-performance-tuning/>
- Pragma Statements*. (2023, 02 16). Retrieved from [sqlite](https://www.sqlite.org/pragma.html): <https://www.sqlite.org/pragma.html>
- RTIC*. (2023). Retrieved from thedatacity: <https://thedatacity.com/rtics/>
- sqlitebrowser*. (2023, 06 14). Retrieved from github: <https://github.com/sqlitebrowser/sqlitebrowser/wiki>
- Standard Industrial Classification*. (2023, August 09). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Standard_Industrial_Classification
- Strengthening new norms in beneficial ownership transparency in Asia and the Pacific region*. (2022, 03 07). Retrieved from Open Ownership: <https://www.openownership.org/en/blog/strengthening-new-norms-in-beneficial-ownership-transparency-in-asia-and-the-pacific-region/>
- The Data City. (2023). *Platform*. Retrieved 08 28, 2023, from <https://thedatacity.com/platform/>

Uehlinger, S. (2023, July 24). *Connecting hidden relationships in shell companies using the Open Ownership Register and Black Ice's SARA*. Retrieved from <https://www.openownership.org/en/blog/connecting-hidden-relationships-in-shell-companies-using-the-open-ownership-register-and-black-ices-sara/>