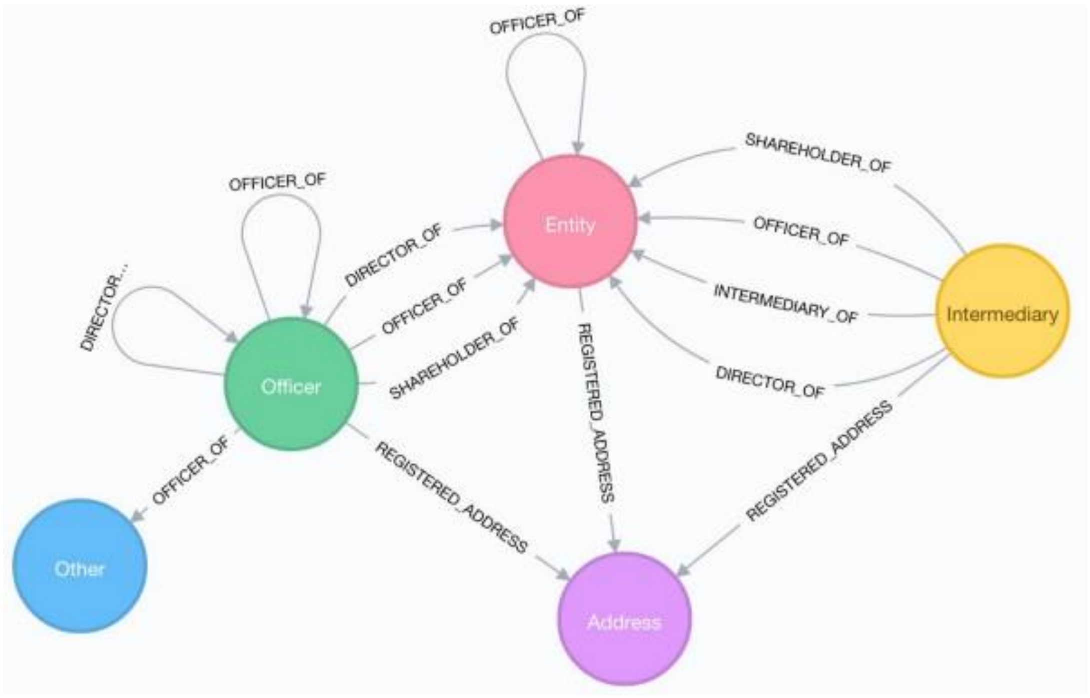
Graph Visualization of Panama Papers Data In Neo4j

Revisiting ICIJ's Offshore Leaks In The Face Of The Latest Deutsche Bank Scandal

William Lyon; Dec 3 2018

https://medium.com/@lyonwj/graph-visualization-of-panama-papers-data-in-neo4j-9c08ca17039c?mkt_tok=eyJpIjoiTWpJeU9HWTFPV0l4TURNMSIsInQiOiJqM2pRQ0xnbkE5OExOK1p0cWZHVlR0OWdYQ2Rtenloblg0QVZEWUcxaU1jMElXck94VzBwNXE3ZU9pbm9DY2JSWW1LOTRIWGVlOVZGXC9cL1FhR3lhTjUxOGJJYXg2OEJJdWtLamw0akJ5TXZOdzZuV05UT0ZcL1RjNHBwUUhNNFBsWiJ9



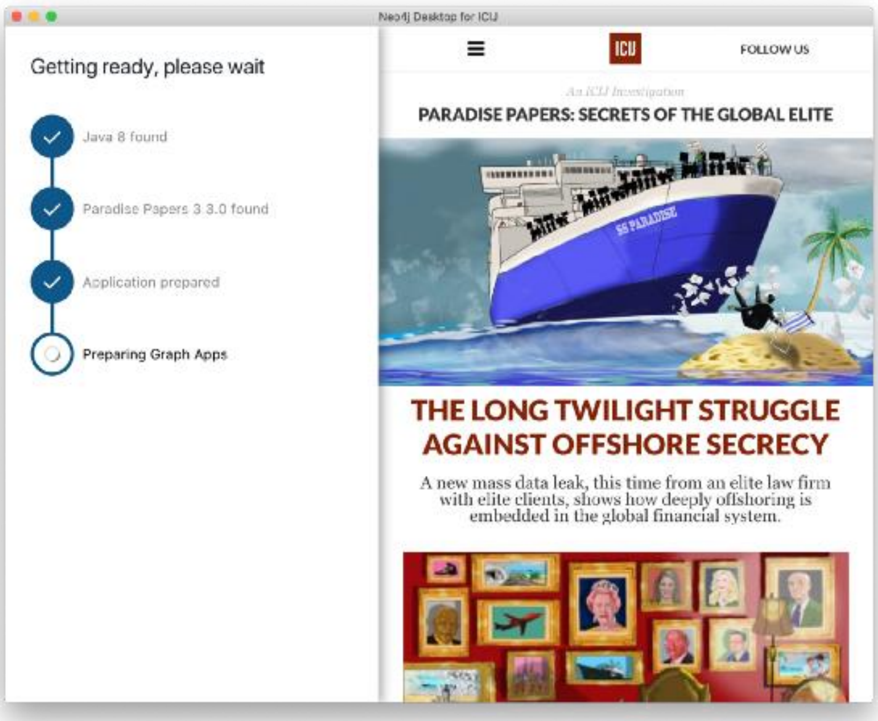
The Panama Papers / Offshore Leaks Neo4j database data model. The database contains offshore legal entities and connections to them.

News last week that the headquarters of Deutsche Bank was raided in a money laundering investigation as a result of the Panama Papers investigation has rekindled interest in the public Neo4j dataset that ICIJ released as part of the Offshore Leaks database. In this post, we take a look at how we can explore the Panama Papers data in Neo4j, in the context of the recent Deutsche Bank investigation.

First, we'll see how we can access the dataset in Neo4j, using either Neo4j Desktop or Neo4j Sandbox. Then we take a look at some simple Cypher queries we can use to search for offshore companies connected to Deutsche Bank, and how to visualize the results, both in Neo4j Browser and Neo4j Bloom. Finally, we learn to use virtual nodes and relationships to simplify the graph visualizations.

We've previously seen some great examples of how to analyze the Offshore Leaks database in Neo4j, so if you're not familiar with the dataset I'd suggest you first give some of those a quick read. They're quite interesting, showing how to create geospatial data visualizations, even showing the shortest path from Rex Tillerson to the Queen of England in the dataset.

To follow along, the first step is to access the Offshore Leaks Neo4j database. This database can either be downloaded from the ICIJ website as "Neo4j Desktop For ICIJ" or accessed via Neo4j Sandbox without any download.



Getting ready, please wait

Java 8 found

Paradise Papers 3.3.0 found

Application prepared

Preparing Graph Apps

PARADISE PAPERS: SECRETS OF THE GLOBAL ELITE

THE LONG TWILIGHT STRUGGLE AGAINST OFFSHORE SECRECY

A new mass data leak, this time from an elite law firm with elite clients, shows how deeply offshoring is embedded in the global financial system.

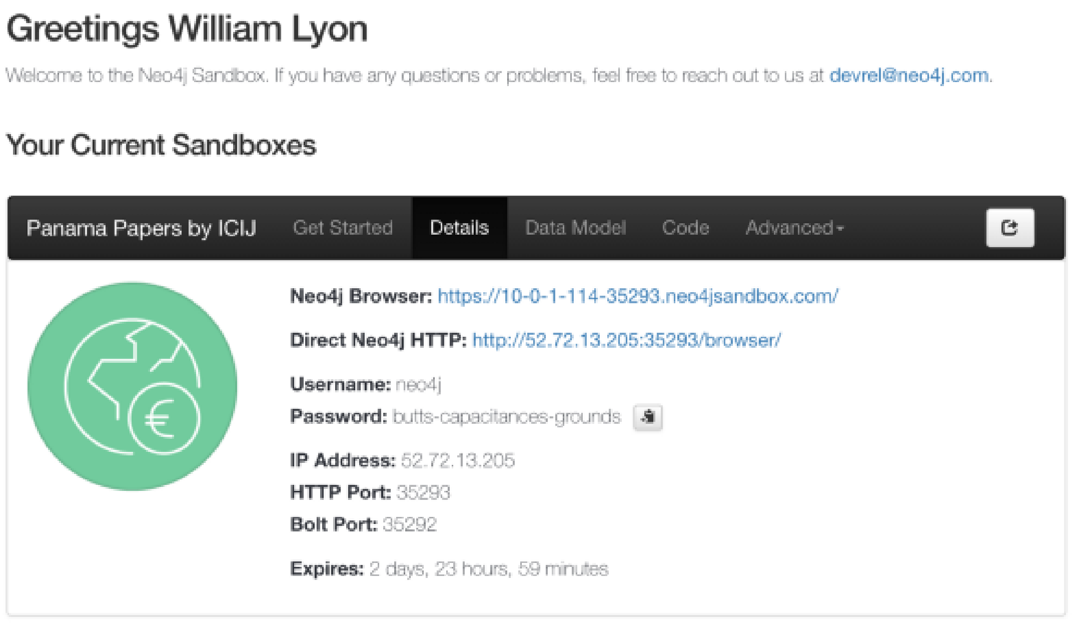
Neo4j Desktop For ICIJ

The ICIJ has made the Offshore Leaks database (which includes Panama Papers and Paradise Papers) available as a downloadable Neo4j database.

The database includes an interactive Neo4j Browser Guide that embeds images, database queries, and text to walk you through exploring the dataset. You can see a screencast here of how it all works.

Neo4j Sandbox

Neo4j Sandbox allows you to spin up Neo4j instances on demand hosted in the cloud. You can choose from several different sandbox "use cases" that include datasets and guides to walk you through querying and visualizing the data.



Greetings William Lyon

Welcome to the Neo4j Sandbox. If you have any questions or problems, feel free to reach out to us at devrel@neo4j.com.

Your Current Sandboxes

Neo4j Browser: https://10-0-1-114-35293.neo4jsandbox.com/

Direct Neo4j HTTP: http://52.72.13.205:35293/browser/

Username: neo4j
Password: butts-capacitances-grounds

IP Address: 52.72.13.205
HTTP Port: 35293
Bolt Port: 35292

Expires: 2 days, 23 hours, 59 minutes

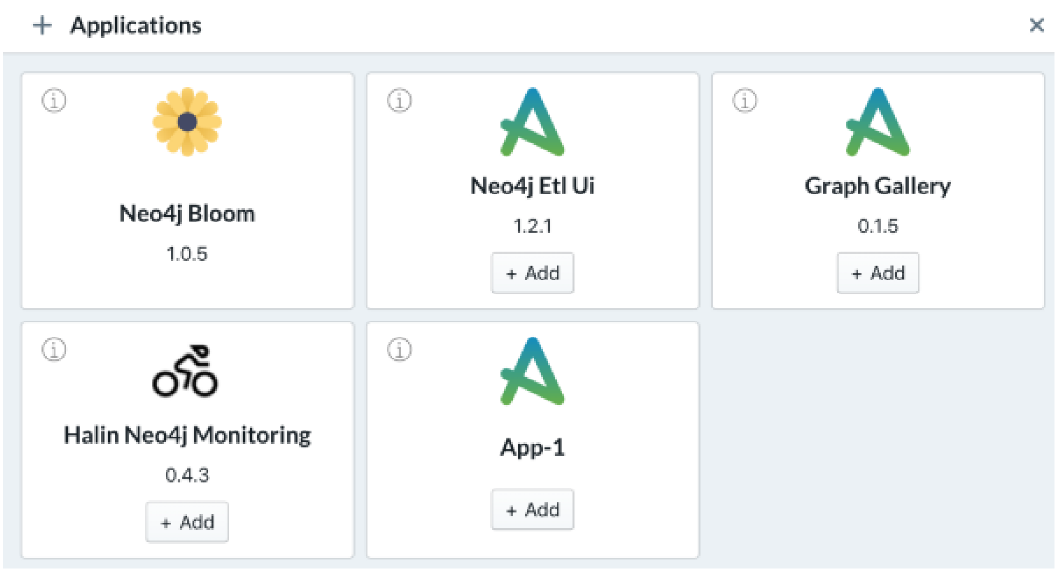
Connection credentials for a Neo4j Sandbox instance.

After signing in to Neo4j Sandbox you'll see "Panama Papers by ICIJ" as one of the sandbox use cases to choose from. Choose "Launch Sandbox" and after a few seconds, your personal Neo4j instance loaded with the Panama Papers dataset will be available and ready to start querying.

Previous posts have shown how to explore the data in Neo4j Browser using Cypher, so instead of rehashing those techniques, we'll jump straight into visualizing the data using Neo4j Bloom.

Graph Visualization With Bloom

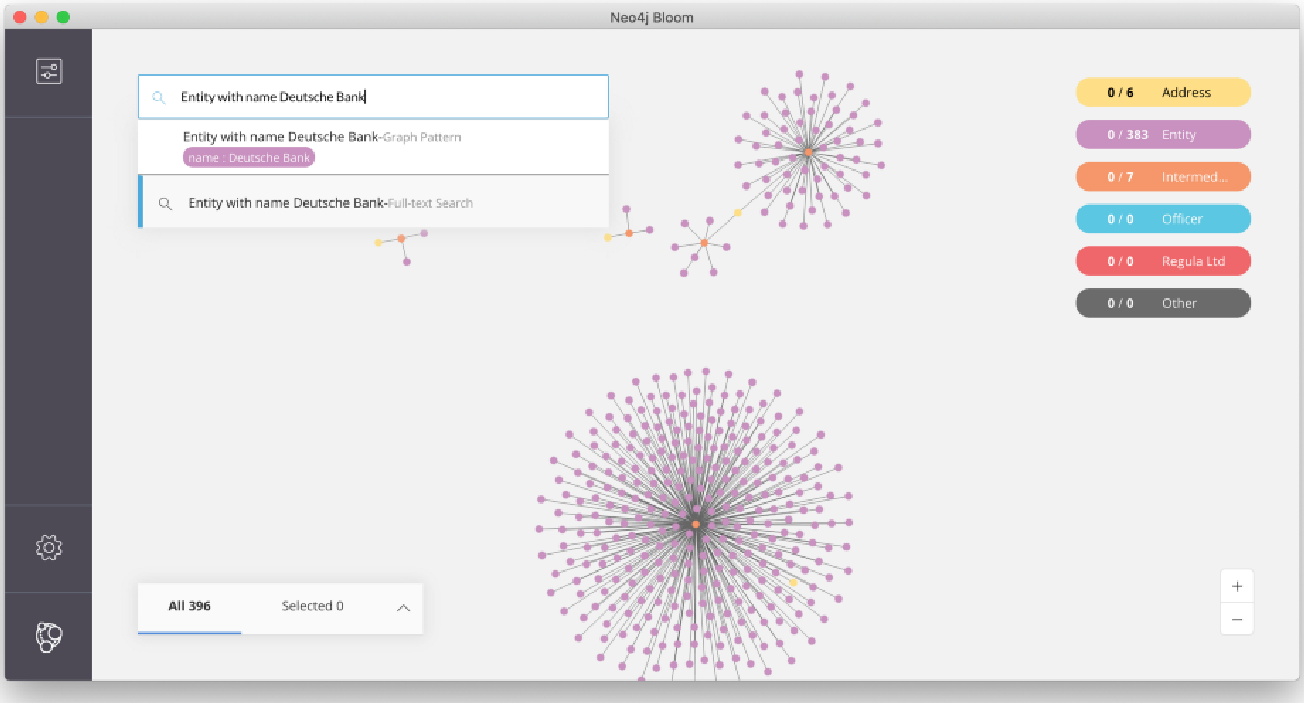
Neo4j Bloom is a graph exploration application for visually interacting with graph data. Bloom is designed to enable exploration of the graph without writing Cypher. Bloom is available through Neo4j Desktop, but you'll need an access key to install it. You can request an activation key here. Once you have your activation key it can be installed the way other Graph Apps are installed in Neo4j Browser.



Neo4j Bloom is one of many "Graph Apps" that can be installed via Neo4j Desktop.

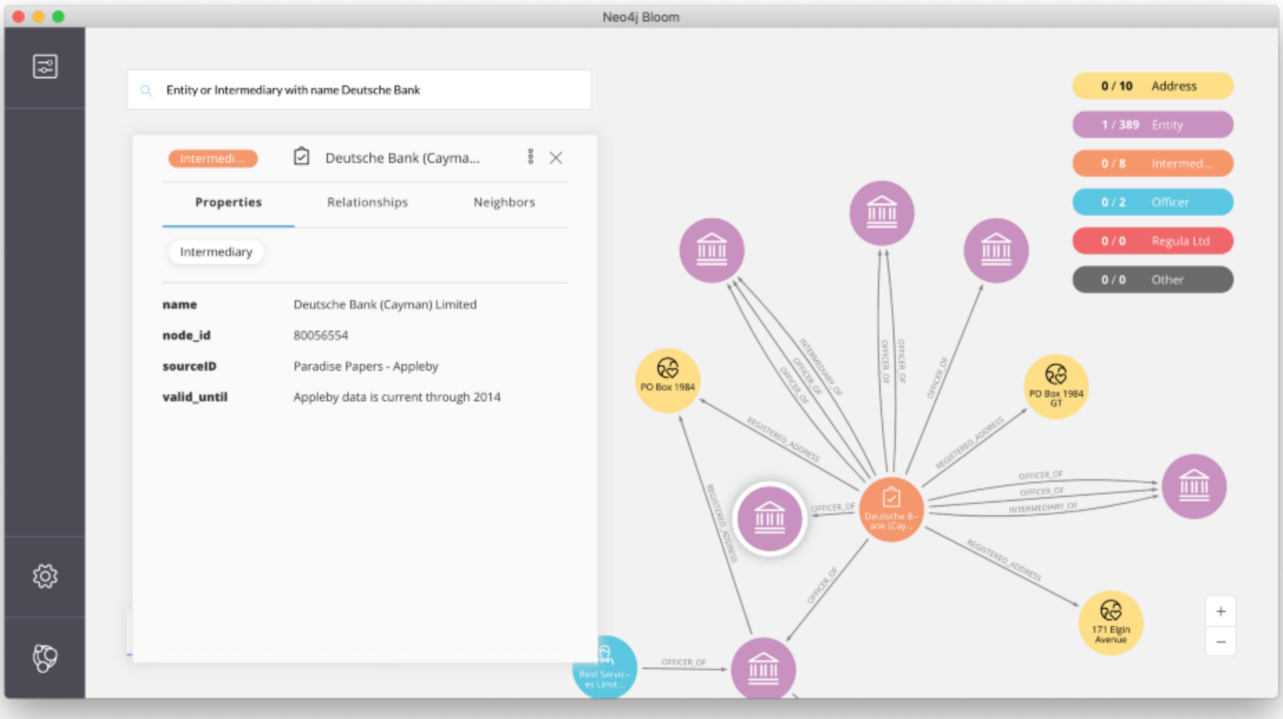
Once Bloom is installed in Neo4j Desktop it will automatically connect to the active Neo4j database. If you're using Neo4j Sandbox you'll need to create a "remote graph" using the connection credentials specific to your private Neo4j sandbox instance.

Bloom uses natural search phrases to query the graph. Here we use a natural search phrase to search for all Entity nodes that contain "Deutsche Bank" in their name. We can then expand out from these nodes to find all connected Entities, Officers, Intermediaries, etc:



Querying the graph in Neo4j Bloom using a natural search phrase.

In Bloom we can also zoom in and inspect the properties of nodes and relationships. Here we inspect one of the Deutsche Bank matches, “Deutsche Bank (Cayman) Limited”. We can see that this entity serves as an officer of several other offshore legal entities, and also served as the “intermediary” (a sort of go-between for creating an offshore legal entity)for several offshore entities:



name	Deutsche Bank (Cayman) Limited
node_id	80056554
sourceID	Paradise Papers - Appleby
valid_until	Appleby data is current through 2014

Inspecting a specific Deutsche Bank offshore legal entity.

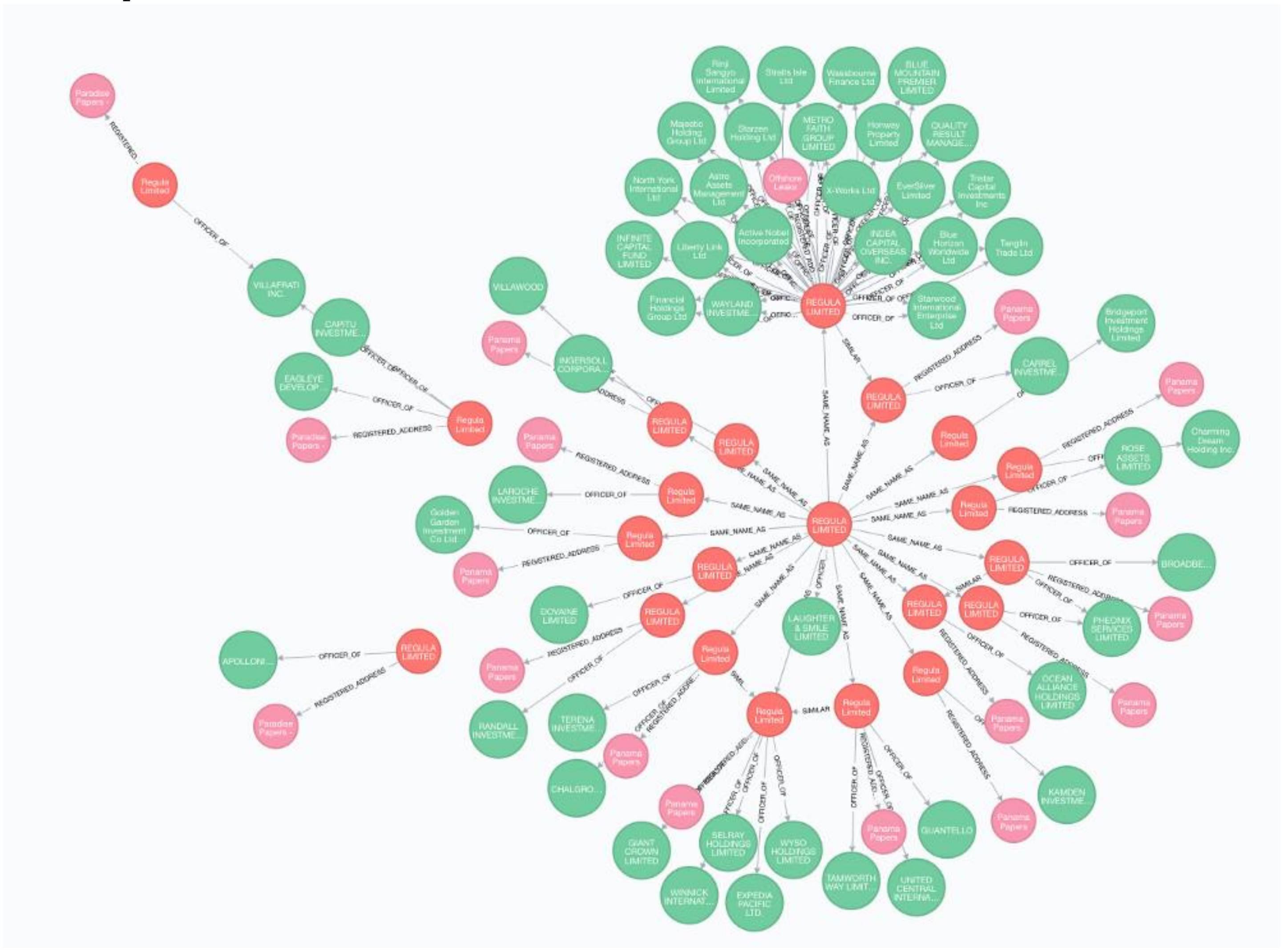
Let’s take a step back from Bloom and see how we can query the dataset using Cypher in Neo4j Browser, using for context some of the recent reports about the current Deutsche Bank investigation.

Querying With Cypher

Recent reporting of the latest Deutsche Bank raids has identified a handful of specific Deutsche Bank subsidiaries that were allegedly involved in the money laundering scheme at the center of the most recent investigations. One of these is “Regula Limited”, an offshore legal entity registered in the Bahamas and British Virgin islands that was connected to Deutsche Bank, as a wholly owned subsidiary.

We can search for Regula Limited and all the companies it is connected to in the Panama Papers dataset with this Cypher query in Neo4j Browser:

```
MATCH path=(regula:Officer)-->()
WHERE toUpper(regula.name) CONTAINS "REGULA LIMITED"
RETURN path
```



Searching for “Regula Limited” and associated connections in the Panama Papers dataset.

One observation we can quickly make is that there are many “Regula Limited” nodes, connected by SAME_NAME_AS relationships. The reason for these seemingly duplicated nodes is that the data came from several sources and the ICIJ was not able to determine with absolute certainty that these are in fact the same legal entities. Obviously, they share a name but given the data available, the ICIJ could not make the assumption that they are the same legal entity. However, for our purposes of exploring the data today we’d like to make the assumption that any legal entity with the name “Regular Limited” is referring to the same legal entity.

Cases like this, where we want to collapse a group of nodes into a single node, including the relationships of the original nodes, are an example of *graph simplification.* The idea of graph simplification is that we have some internal representation of the data in the database, but we want to present a simplified representation of the data to the user for visualization. One way to accomplish this simplification is through the use of virtual nodes and relationships.

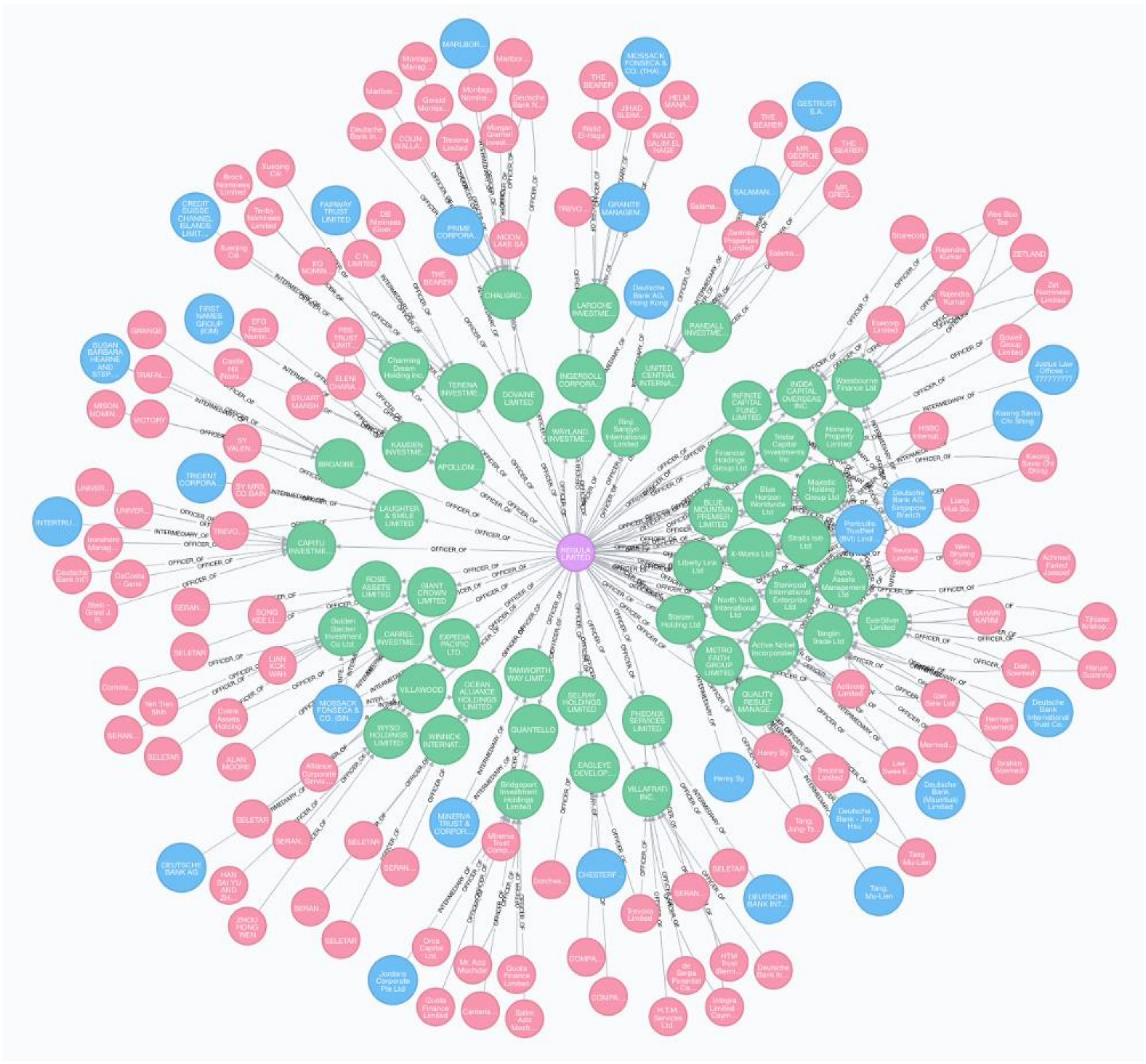
Graph Simplification With Virtual Nodes And Relationships

Virtual nodes and relationships are graph objects that do not exist in the database, but can be constructed at query time to be used by tools like Neo4j Browser or Bloom for visualization. We can create virtual nodes using the APOC procedure library and the functions apoc.create.vNode and apoc.create.vRelationship

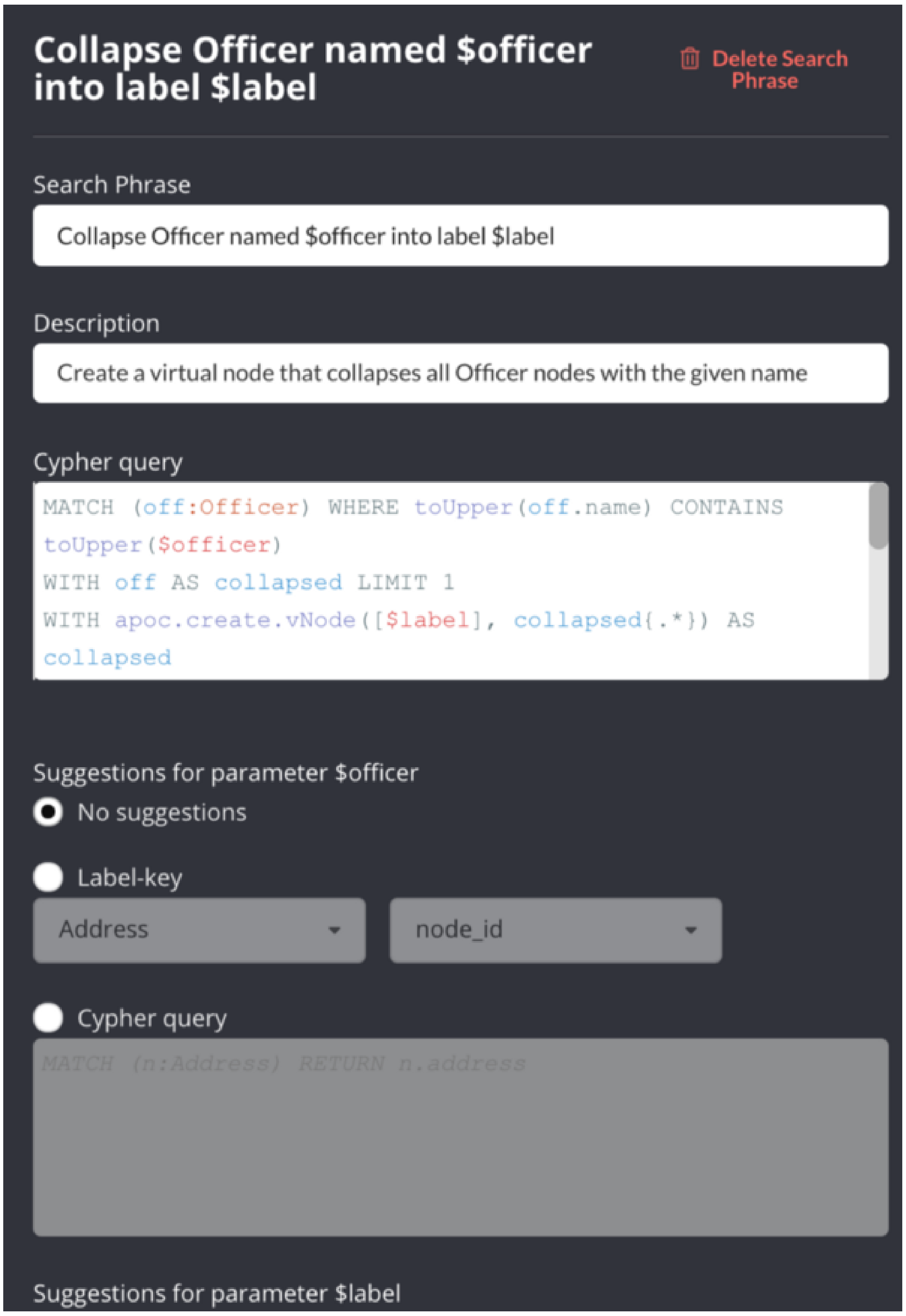
Here’s an example showing how we can collapse all “Regula Limited” nodes into a single node, including all relationships, using virtual nodes and relationships:

```
MATCH (off:Officer)
WHERE toUpper(off.name) CONTAINS "REGULA LIMITED"
WITH off AS collapsed LIMIT 1
WITH apoc.create.vNode(["Regula"], collapsed {.*}) AS collapsed
MATCH (off)-[col1]->(e:Entity)
WHERE toUpper(off.name) CONTAINS "REGULA LIMITED"
WITH collapsed, off, e,
  apoc.create.vRelationship(collapsed, type(col1), {}, e) AS r1
MATCH (e)<-[r2]-(o2)
WHERE (o2:Officer OR o2:Intermediary)
  AND NOT toUpper(o2.name) CONTAINS "REGULA LIMITED"
RETURN collapsed, e, r1, r2, o2
```

Running this query in Neo4j Browser gives us a simplified view of the data. Now we can more easily see the connections to “Regula Limited”:


Visualizing Deutsche Bank subsidiary, Regula Limited, as a single node in the graph using virtual nodes and relationships.

Virtual Nodes and Relationships In Bloom With Search Phrases



Collapse Officer named $officer into label $label

Delete Search Phrase

Search Phrase

Collapse Officer named $officer into label $label

Description

Create a virtual node that collapses all Officer nodes with the given name

Cypher query

```
MATCH (off:Officer) WHERE toUpper(off.name) CONTAINS
toUpper($officer)
WITH off AS collapsed LIMIT 1
WITH apoc.create.vNode([$label], collapsed{.*}) AS
collapsed
```

Suggestions for parameter $officer

[✓ No suggestions]

Label-key

Address node_id

Cypher query

```
MATCH (n:Address) RETURN n.address
```

Suggestions for parameter $label

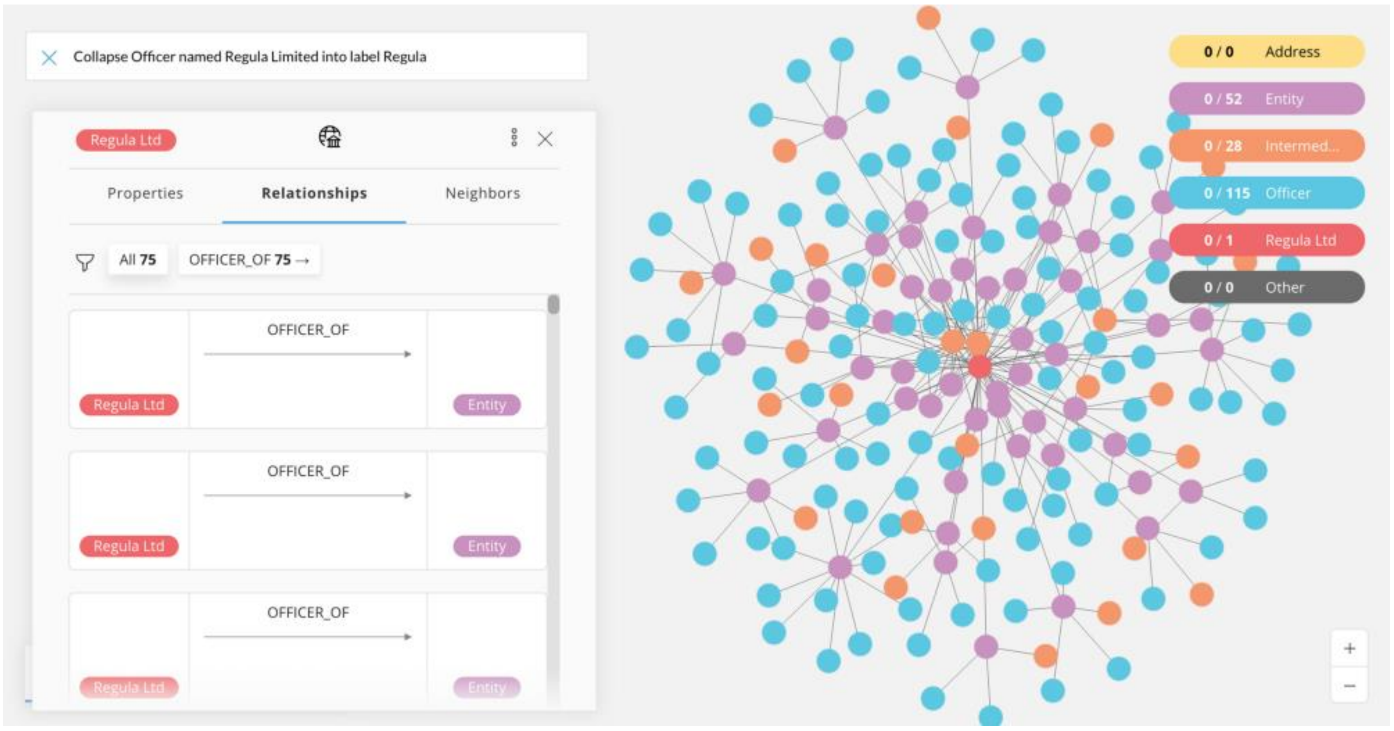
Creating a Search Phrase in Bloom to use virtual nodes and relationships in our graph visualization.

We saw how to simplify our graph visualization with virtual nodes and relationships in Neo4j Browser, but how can we use the same technique in Bloom? To create virtual nodes and relationships in Bloom we will make use of a feature in Bloom called Search Phrases.

Search Phrases allow us to define a parameterized Cypher query that can be used as part of the natural language search in Bloom, with user provided values for the parameters at search time. For example, we will create a search phrase

```
Collapse Officer named $officer into label $label
```

that maps to the Cypher query we used above for creating virtual nodes and relationships for Regular Limited. But because we parameterize it, the search phrase can be used to collapse any entity with inputs specified by the user. Here’s how we use it in Bloom:



Using a search phrase in Bloom to collapse all Regula Limited nodes into a single node in the visualization.

Here are some resources to help explore the data further:

Resources

- Learn more about Bloom features in the docs. You can also register your interest in Bloom here.
- The Offshore Leaks Database by ICJI

- Neo4j
- Data Visualization
- Database