

# Graphen: Neo4j für Korrespondenzdaten

Workshop im Rahmen der Veranstaltungsreihe "Digital Humanities – Wie geht das?"  
des Referats für Digitale Forschungsdienste

Dr. Franziska Klemstein | 27.02.2023

# **Agenda**

## **Graphen: Neo4j für Korrespondenzdaten**

1. Vorstellungsrunde

2. Knoten und Kanten – Erste Schritte

– 10.30 Uhr Pause –

3. Datenimport mit csv-Dateien

4. Erweiterungen, Ergänzungen und Ausblick

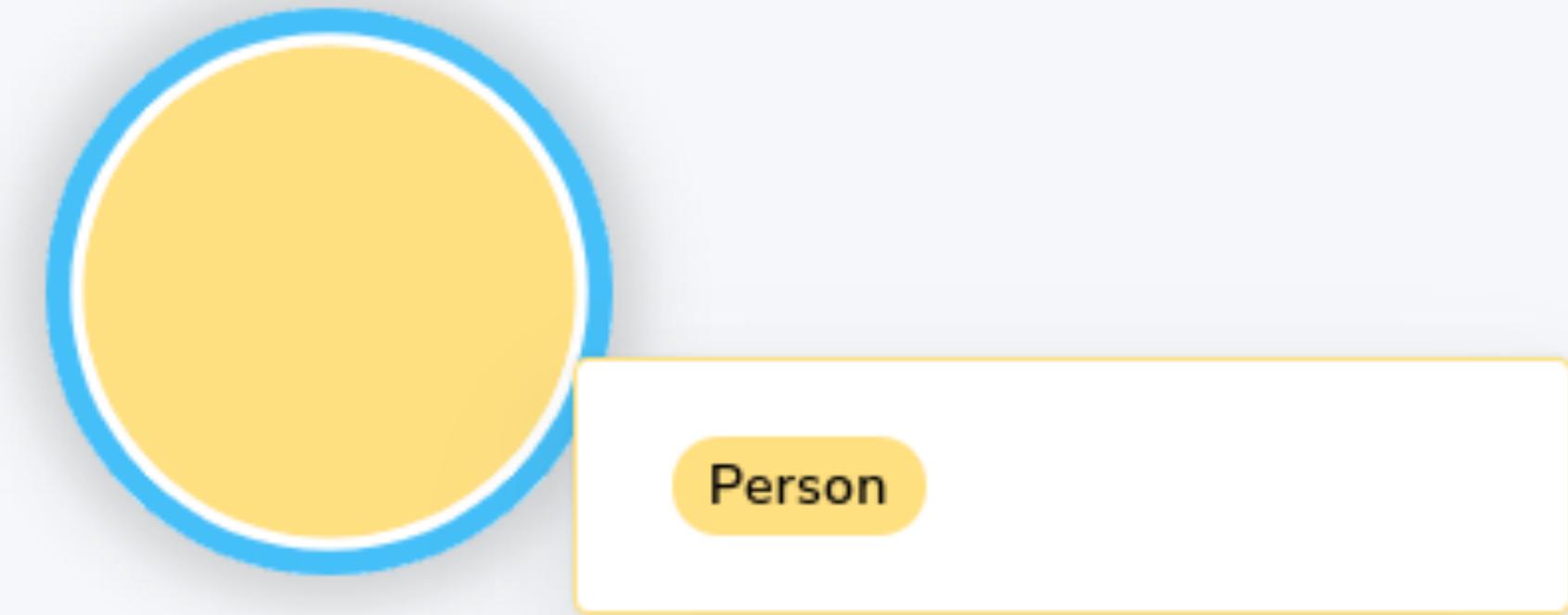
# Vorstellungsrunde

## Interessen und Vorkenntnisse



# Knoten und Kanten

## Erste Schritte



# Vorbereitung

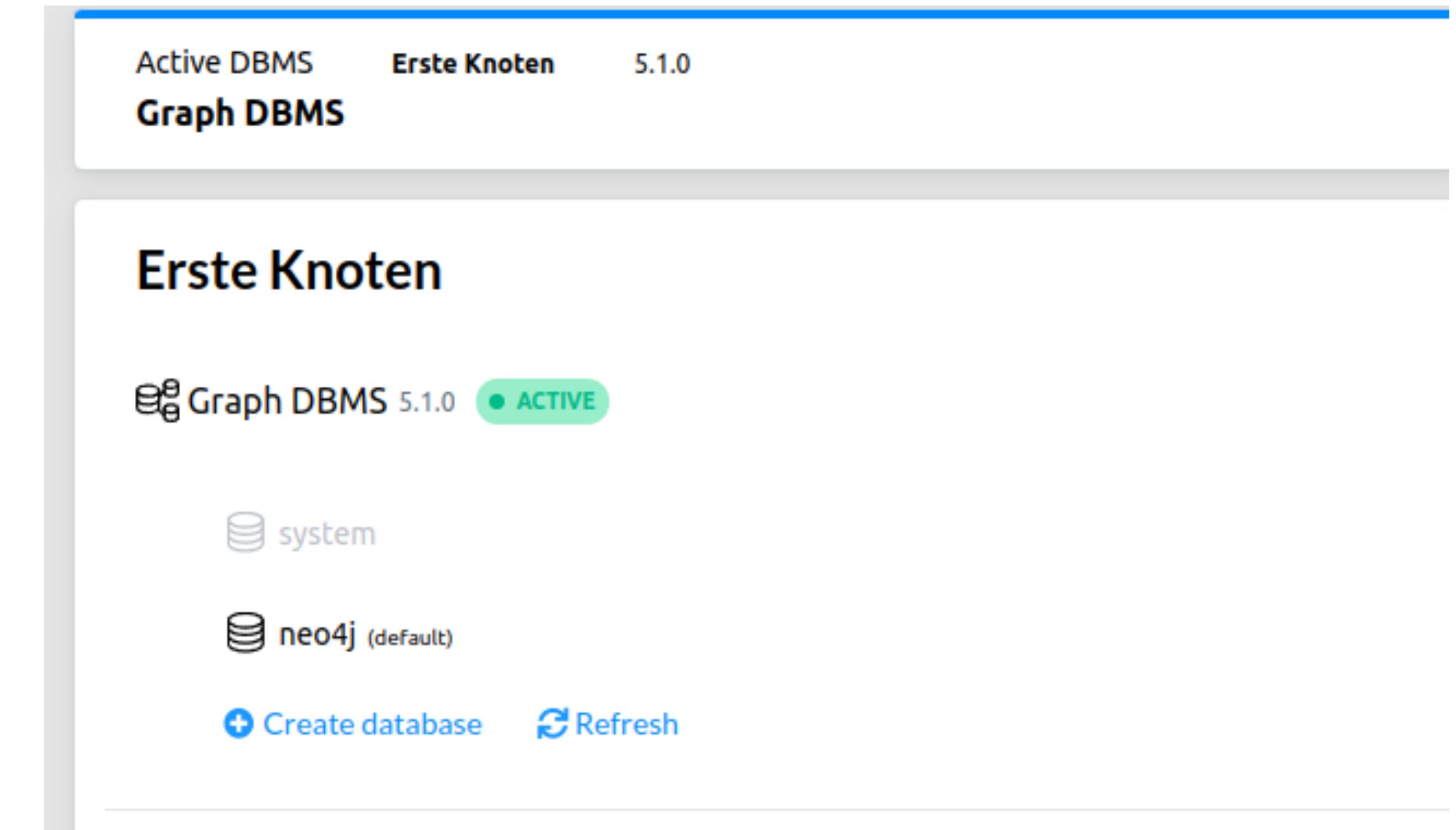
## Startpunkt

- Download und Installation von neo4j

ODER

- neo4j-Registrierung im Browser:

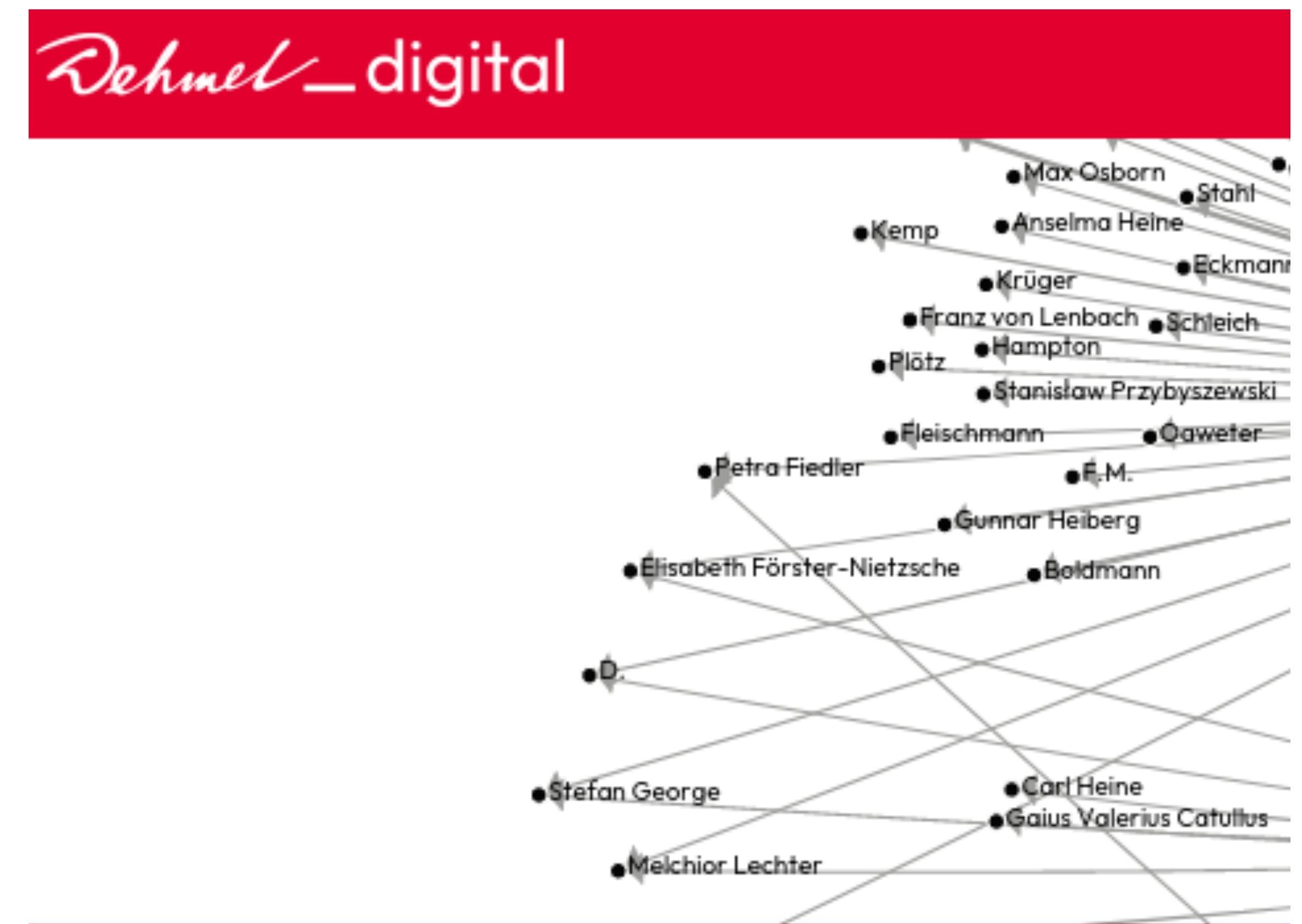
[console.neo4j.io/](http://console.neo4j.io/)



# Beispielgraph

## mit Daten von Dehmel-Digital

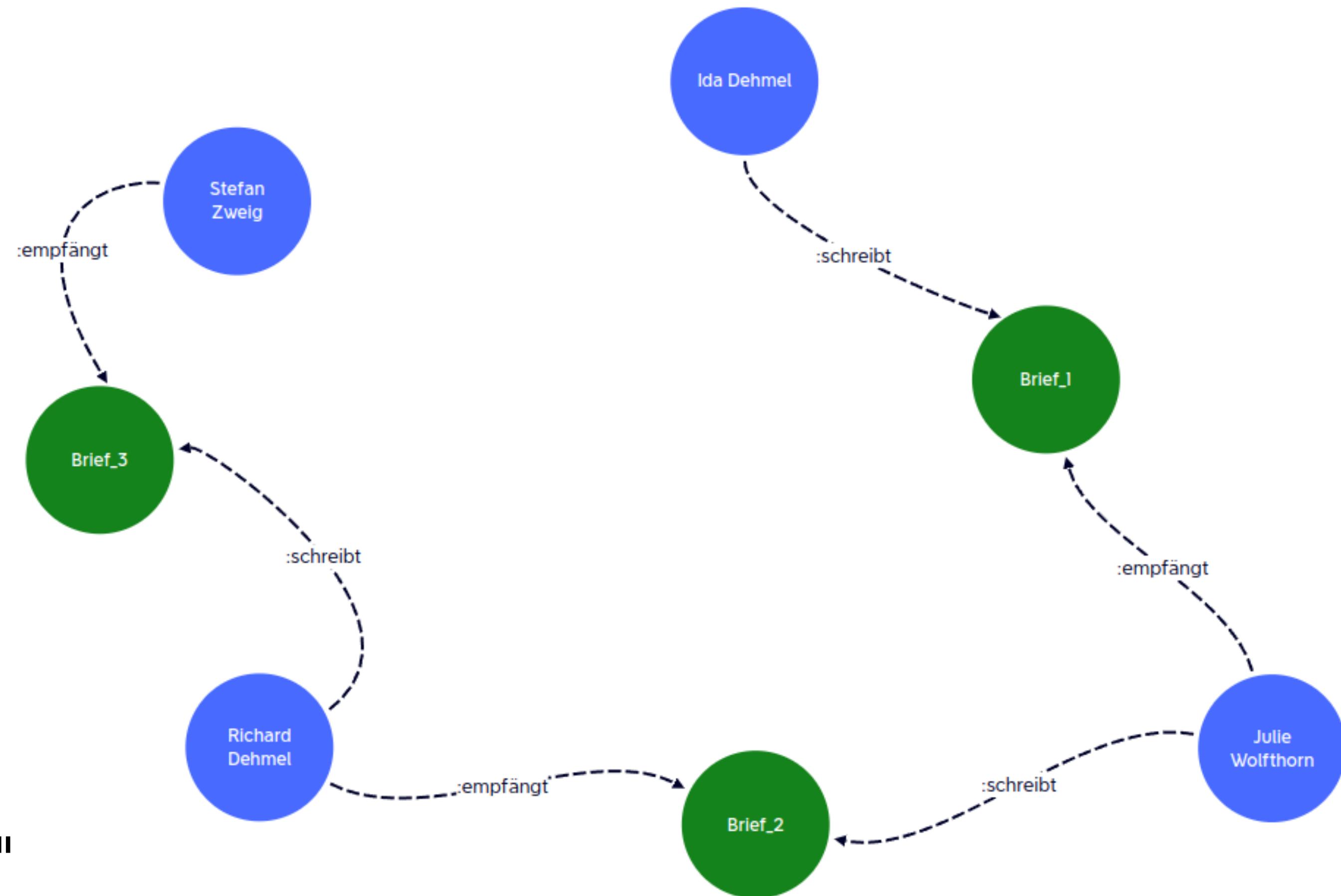
- Knoten und Knotentypen
- mit Labeln und Eigenschaften
- Kanten / Verbindungen / Relationen



# Beispielgraph

## Ein erstes Modell

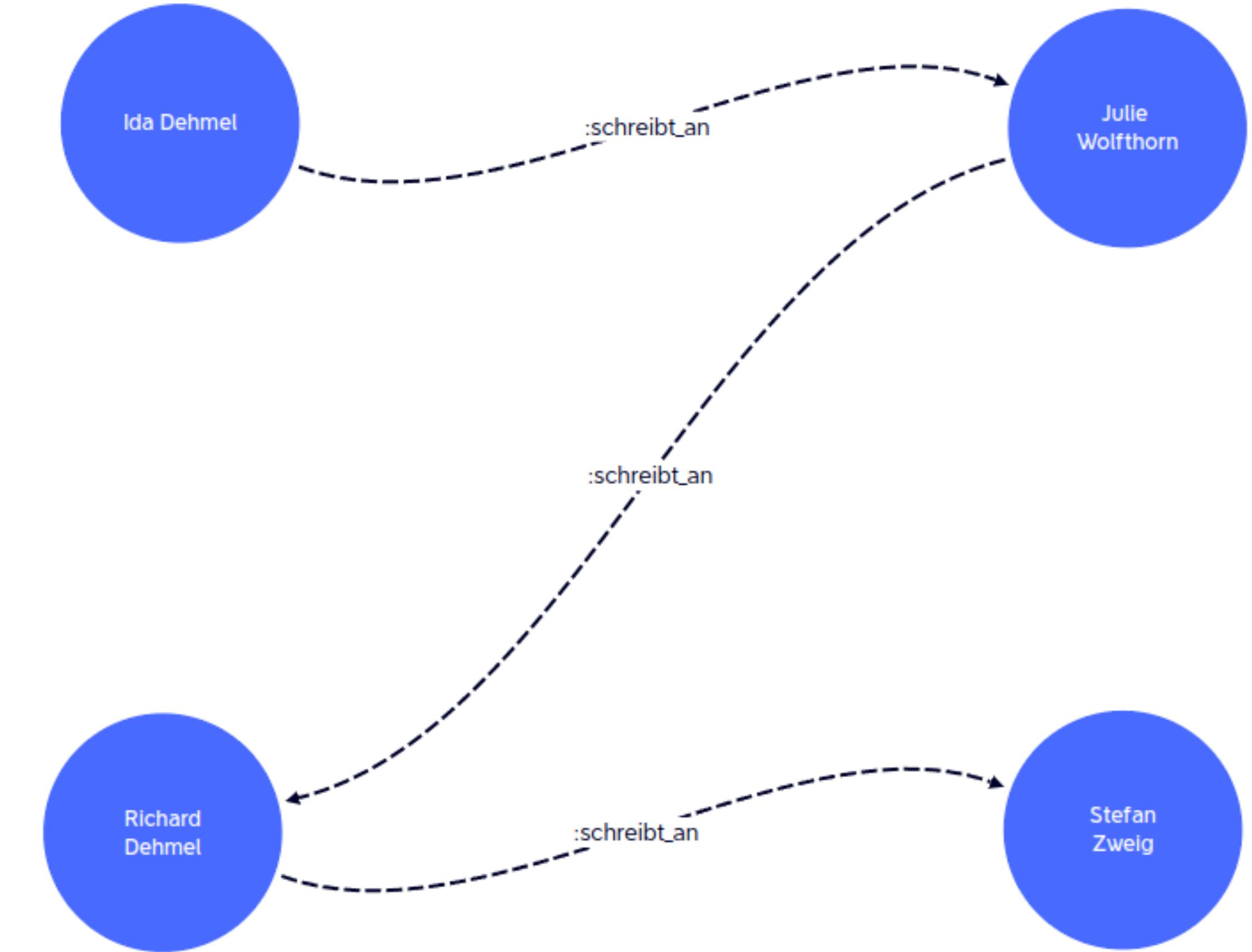
- Knotentyp: Person, Brief
- Label: Person, Brief
- Eigenschaften: Name, Bezeichnung
- Kantentypen: "schreibt", "empfängt"
- Kanteneigenschaften; "keine"



# Beispielgraph

## Ein zweites Modell

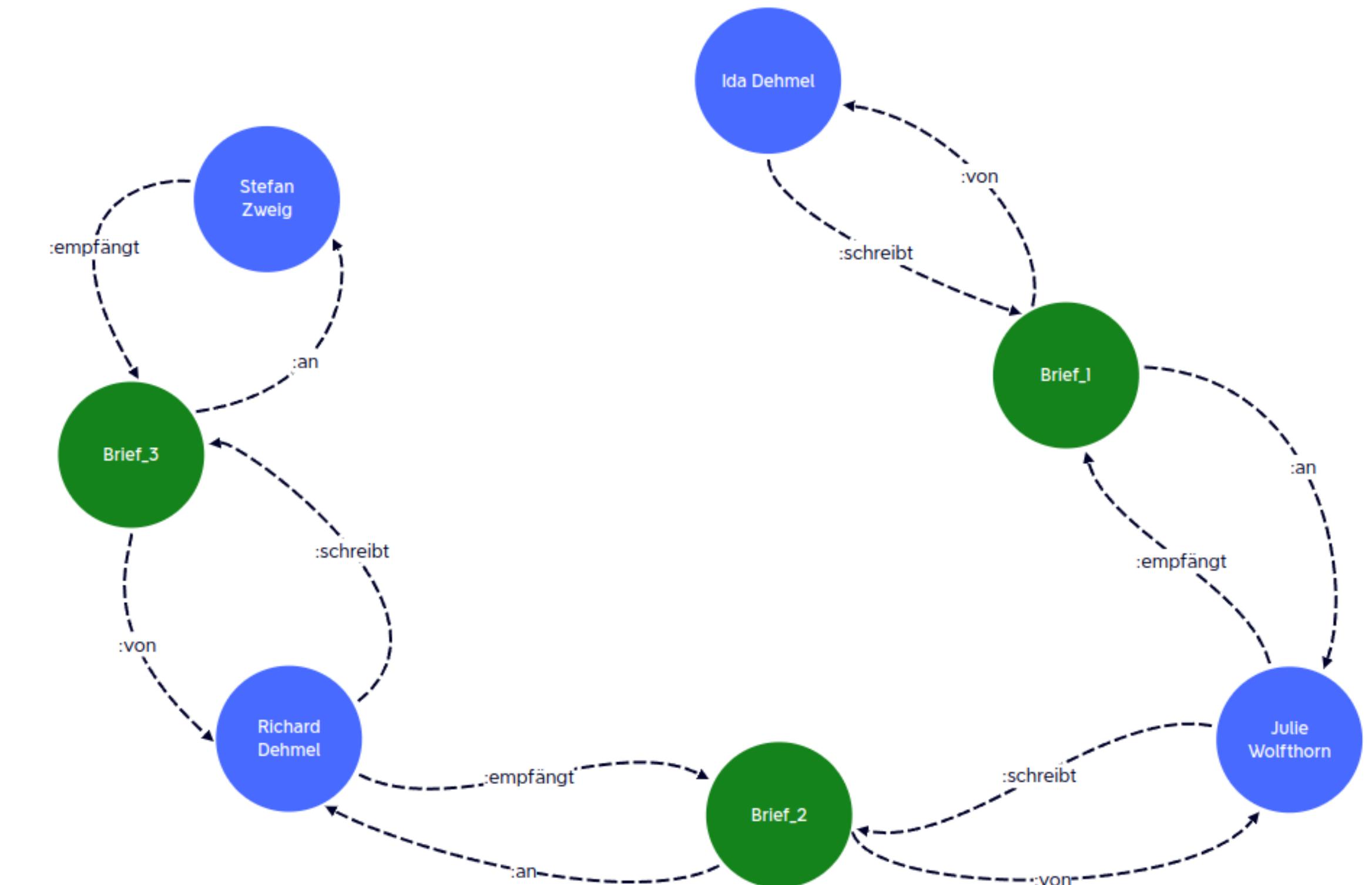
- Knotentyp: Person
- Label: Person
- Eigenschaften: Name
- Kantentyp: "schreibt\_an"
- Kanteneigenschaften: keine



# Beispielgraph

## Ein drittes Modell

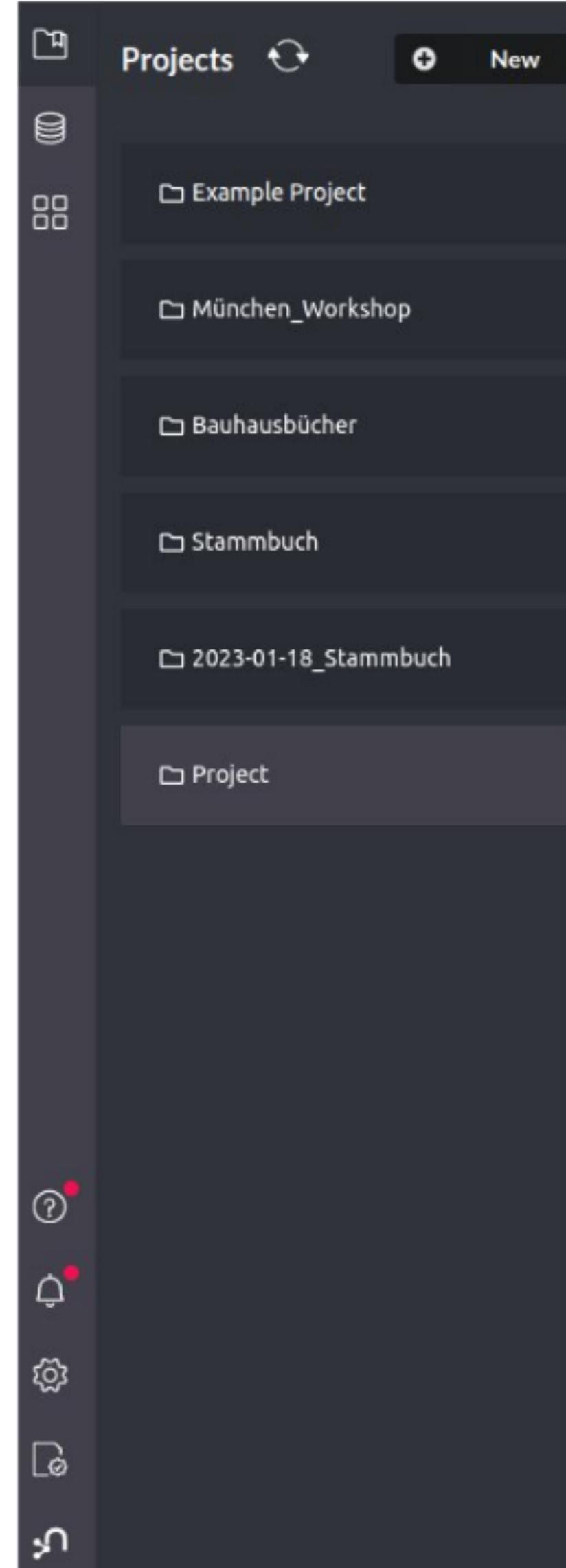
- Knotentyp: Person, Brief
- Label: Person, Brief
- Eigenschaften: Name, Bezeichnung
- Kantentypen: "schreibt", "empfängt", "an", "von"
- Kanteneigenschaften: keine



A large hot air balloon is shown from a low angle, filling most of the frame. The balloon's envelope is a vibrant rainbow of colors, starting with red on the left and transitioning through orange, yellow, green, and blue towards the right. The surface of the envelope is visible with its internal ribbing and stitching. The background is a clear, pale blue sky.

Let's get started

File Edit View Window Help Developer



No active DBMS

### Project

Add a DBMS to get started.

---

### File

Add project files to get started.

[Reveal files in File Manager](#) [Filename ▾](#)

This main workspace is titled 'Project'. It displays two sections: 'DBMS' and 'File'. The 'DBMS' section is currently inactive, showing the message 'Add a DBMS to get started.' The 'File' section is also inactive, showing the message 'Add project files to get started.' At the bottom right of the 'File' section, there are two buttons: 'Reveal files in File Manager' and 'Filename ▾'.

Projects    New

- Example Project
- München\_Workshop
- Bauhausbücher
- Stammbuch
- 2023-01-18\_Stammbuch
- New-Project

No active DBMS

## New-Project

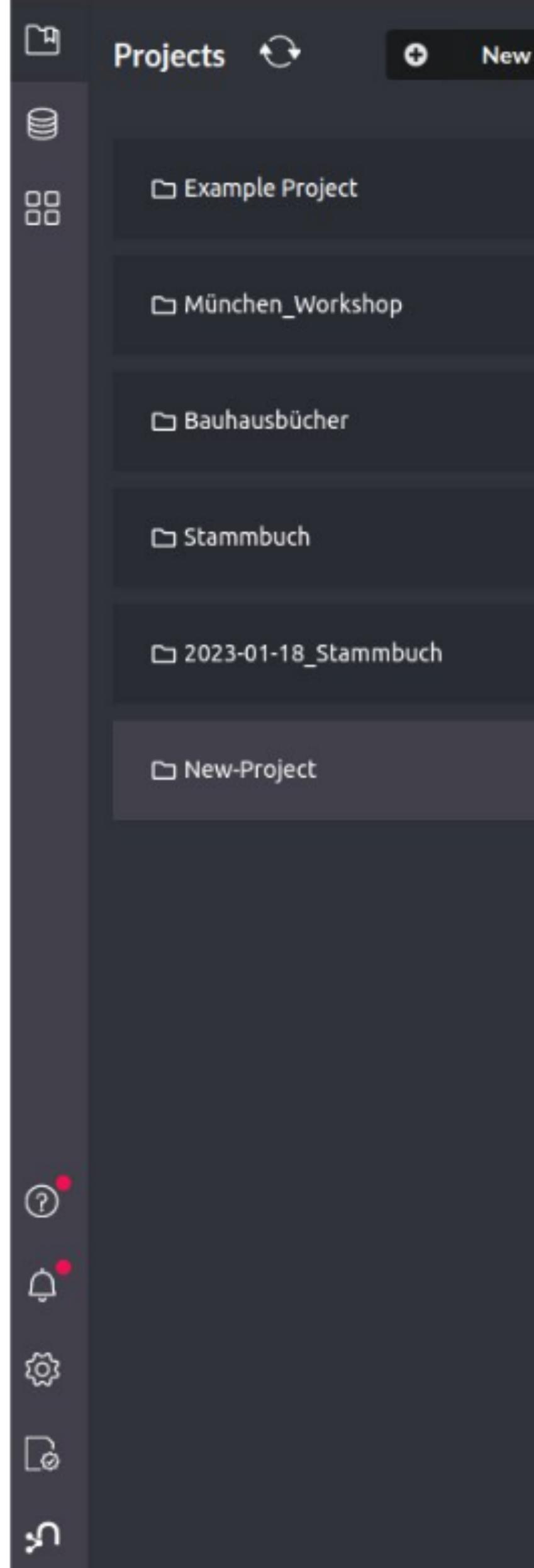
Add a DBMS to get started.

Add project files to get started.

File

Reveal files in File Manager    Filename

Local DBMS  
Remote connection



No active DBMS

## New-Project

**Name**  
Graph DBMS

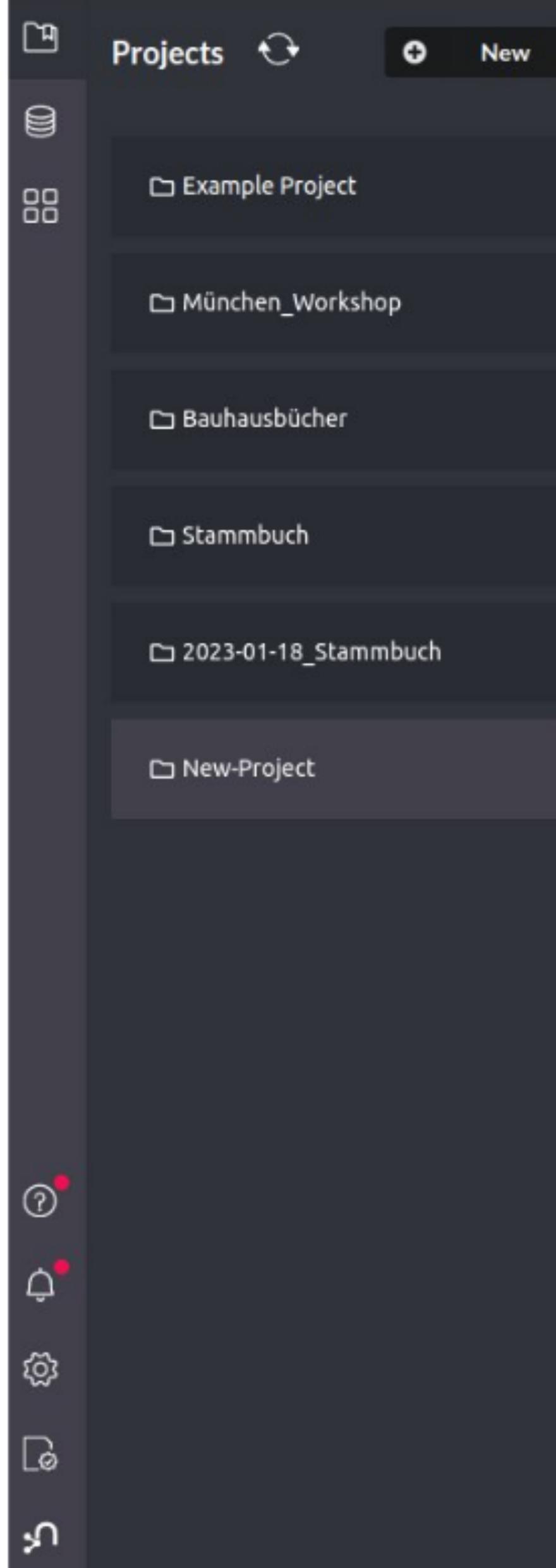
**Password**  
••••

**Version**  
5.1.0

Cancel Create

File Reveal files in File Manager Filename ▾

Add project files to get started.



The left sidebar contains the following items:

- Projects (selected)
- New
- Example Project
- München\_Workshop
- Bauhausbücher
- Stammbuch
- 2023-01-18\_Stammbuch
- New-Project (highlighted with a blue bar)

The bottom toolbar includes:

- Help
- File Manager
- Terminal
- Logs
- Metrics
- Metrics (with a red dot)
- Logs (with a red dot)
- File Manager (with a red dot)
- Terminal (with a red dot)

The top header shows:

- Active DBMS: Graph DBMS
- New-Project
- 5.1.0
- Stop
- Open
- ...
- Search icon

The main area is titled "New-Project" and displays the following:

- Graph DBMS 5.1.0 (ACTIVE)
- system
- neo4j (default)
- Create database
- Refresh

A "File" tab is open in the bottom right corner, showing:

- Reveal files in File Manager
- Filename

The message "Add project files to get started." is displayed.

The sidebar contains several icons: a question mark (help), a gear (settings), and a circular arrow (refresh).

neo4j\$

\$ :play start

**neo4j**

**Getting started with Neo4j Browser**  
Neo4j Browser user interface guide  
[Get started](#)

**Try Neo4j with live data**  
A complete example graph that demonstrates common query patterns.  
Actors & movies in cross-referenced pop culture.  
[Open guide](#)

**Cypher basics**  
Intro to Graphs with Cypher  
What is a graph database?  
How can I query a graph?  
[Start querying](#)

Copyright © Neo4j, Inc 2002-2023

Sign up for a free Neo4j cloud instance with **neo4j aura**

\$ :server status

## Connection status

This is your current connection information.

You are connected as user **neo4j**

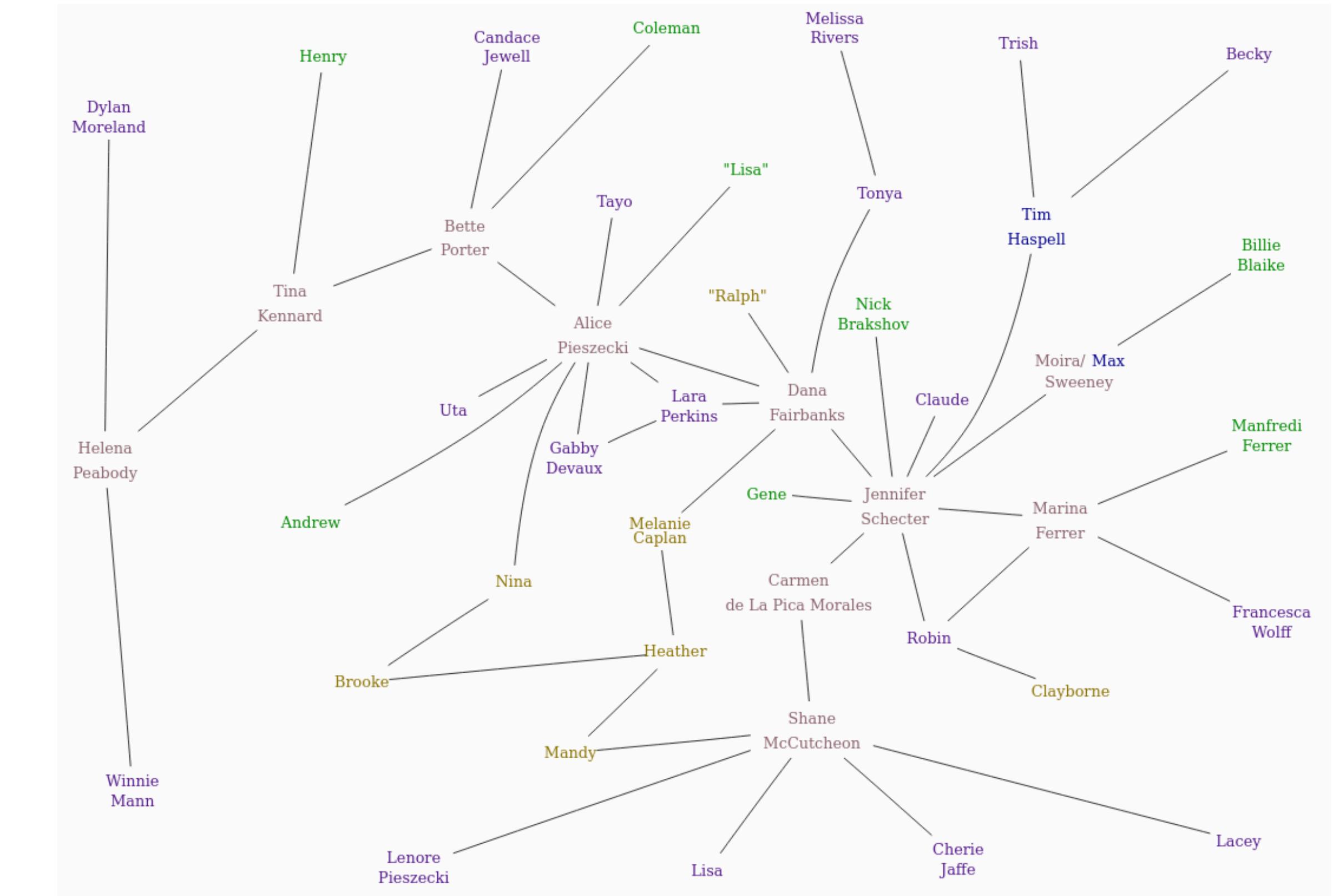
to **bolt://localhost:7687**

Connection credentials are stored in your web browser.

# Graphen

## Relationen, Netzwerke, Verbindungen

- "Die Welt ist ein Graph"  
(Michael Hunger, neo4j)
- Graphentheorie: Leonard Euler
- Graphdatenbanken: Teil der "NoSQL-Bewegung"
- Das Datenmodell besteht aus Knoten, die mittels gerichteter, getypter Verbindungen miteinander verknüpft sind. Beide können beliebige Mengen von Attribut-Wert-Paaren (Properties) enthalten. Daher wird dieses Datenmodell auch als „Property-Graph“ bezeichnet



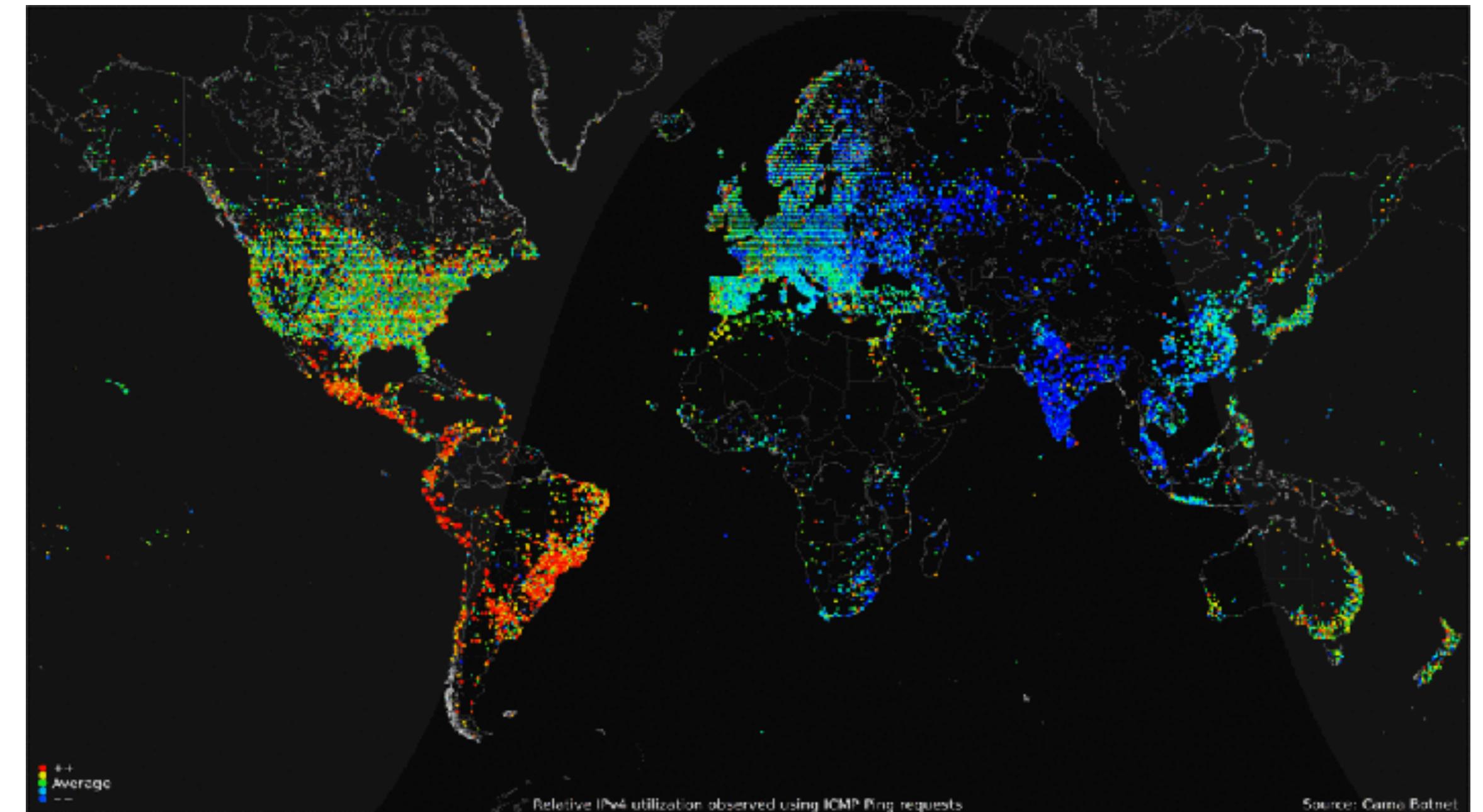
Bildquelle: Silje Ljosland Bakke: The Chart from the TV series The L Word.  
[https://commons.wikimedia.org/wiki/File:TheLWord\\_TheChart.svg?uselang=de#/media/File:TheLWord\\_TheChart.svg](https://commons.wikimedia.org/wiki/File:TheLWord_TheChart.svg?uselang=de#/media/File:TheLWord_TheChart.svg)

# Die Welt ist ein Graph

## Übung 1

Zeichnet / überlegt euch ein Netzwerk:

- Welche Knotentypen mit welchen Eigenschaften sollte es geben?
- Welche Relationen/Kanten?
- Wie sollte der Graph gerichtet sein?
- Was wären mögliche Abfragen?



World map of 24 hour relative average utilization of IPv4 addresses observed using ICMP ping requests

<https://om.co/gigaom/check-out-this-visual-map-that-shows-24-hours-of-internet-usage-around-the-world/>

# Knoten

## Der einfachste mögliche Graph

- Knoten werden verwendet, um Entitäten darzustellen
- Knoten können (mehrere) Label besitzen
- Knoten können Eigenschaften haben

The screenshot shows the Neo4j browser interface. On the left, there is a sidebar with four items: 'Graph' (selected), 'Table', 'Text', and 'Code'. The main area displays a single blue circular node with the number '1' in white inside it. Above the node, the Neo4j command line is visible with the text: 'neo4j\$ CREATE (n:Person) RETURN n;'. The background of the main area is light gray.

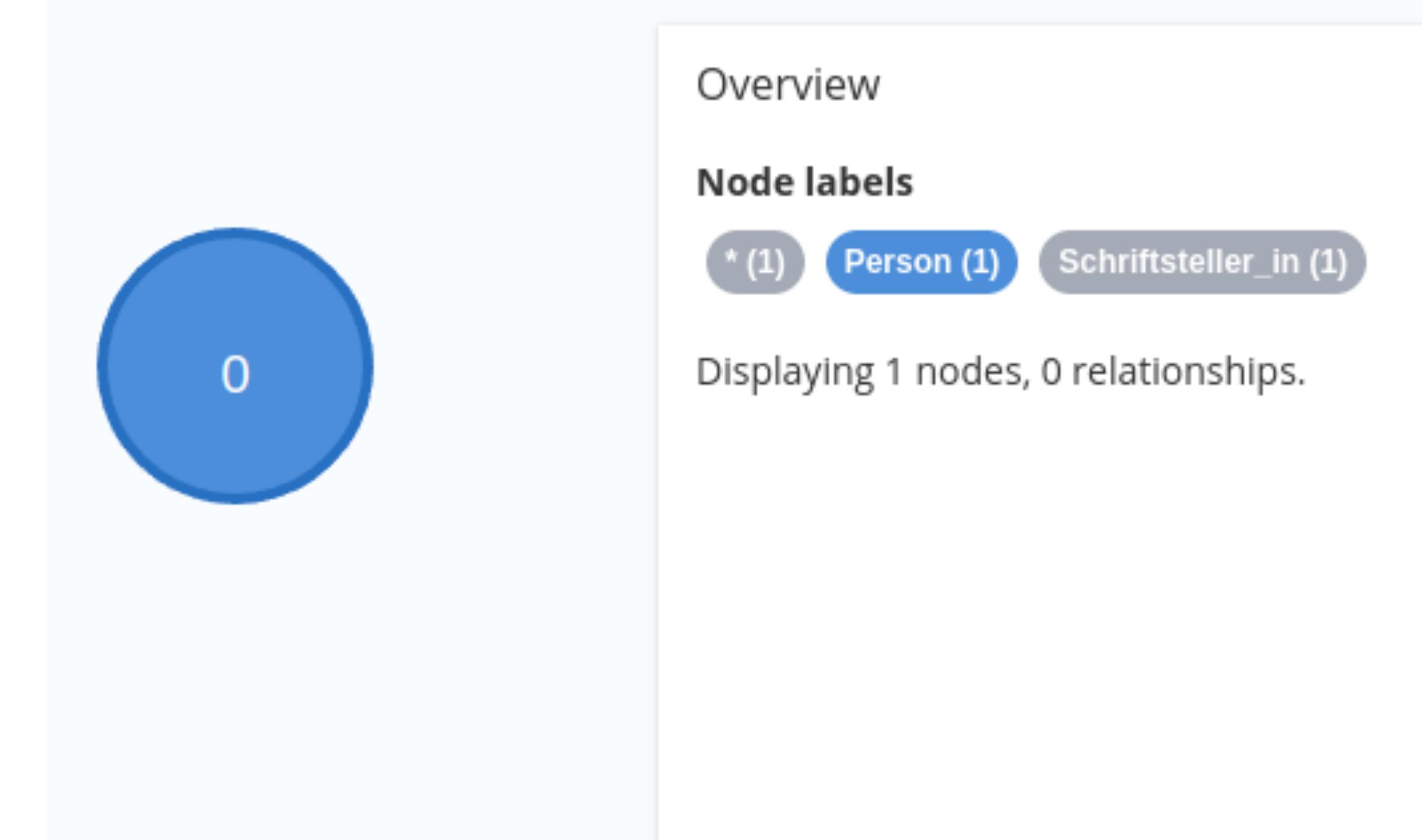
```
neo4j$ CREATE (n:Person) RETURN n;
```

# Knoten mit einem Label

- CREATE (n:Person)  
RETURN n;

## mit mehreren Label

- CREATE (n:Person:Schriftsteller\_in)  
RETURN n;



```
CREATE (:Person {name:'Ida Dehmel', GND:'https://d-nb.info/gnd/118679228'})*
```

```
CREATE (:Person {name:'Ida Dehmel', GND:'https://d-nb.info/gnd/118679228'})
```

Label      Eigenschaft      Eigenschaft

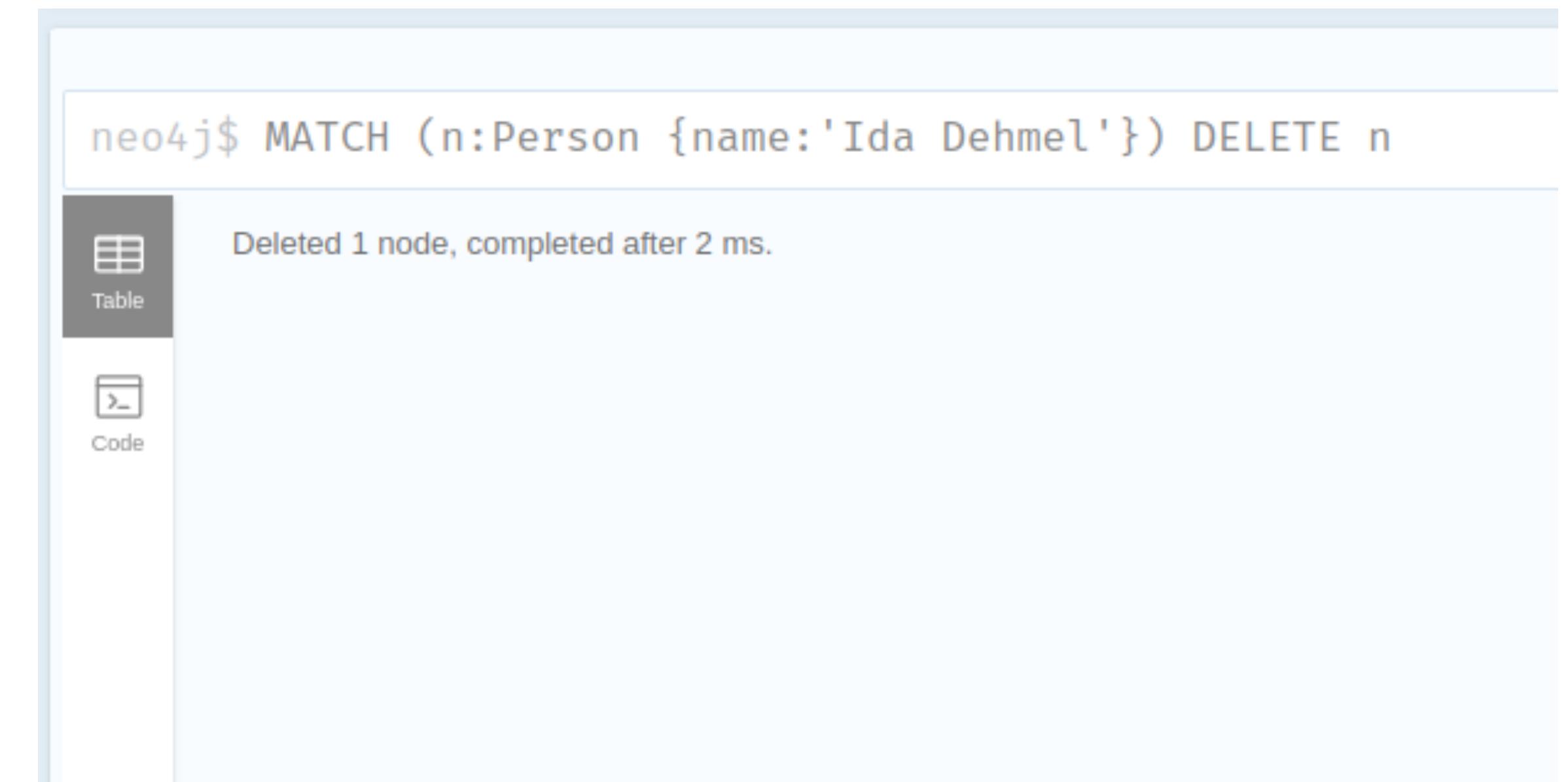
```
RETURN n;
```

\*Knoten mit denselben Label-Bezeichnungen und denselben Eigenschaften können auch mehrfach angelegt werden

Knoten löschen (Option A):

```
MATCH (n:Person {name:'Ida Dehmel'})
```

```
DELETE n;
```



The screenshot shows the Neo4j browser interface. On the left, there is a sidebar with two buttons: 'Table' (which is selected) and 'Code'. The main area contains a command-line interface (CLI) window. The CLI output is as follows:

```
neo4j$ MATCH (n:Person {name:'Ida Dehmel'}) DELETE n
Deleted 1 node, completed after 2 ms.
```

## Knoten löschen (Option B):

```
MATCH (n:Person)  
WHERE ID(n)=0  
DELETE n
```



The screenshot shows a query execution results page from the Neo4j browser. On the left, there is a sidebar with two tabs: 'Table' (selected) and 'Code'. The 'Code' tab shows the Cypher query:

```
1 MATCH (n:Person)  
2 WHERE ID(n)=0  
3 DELETE n
```

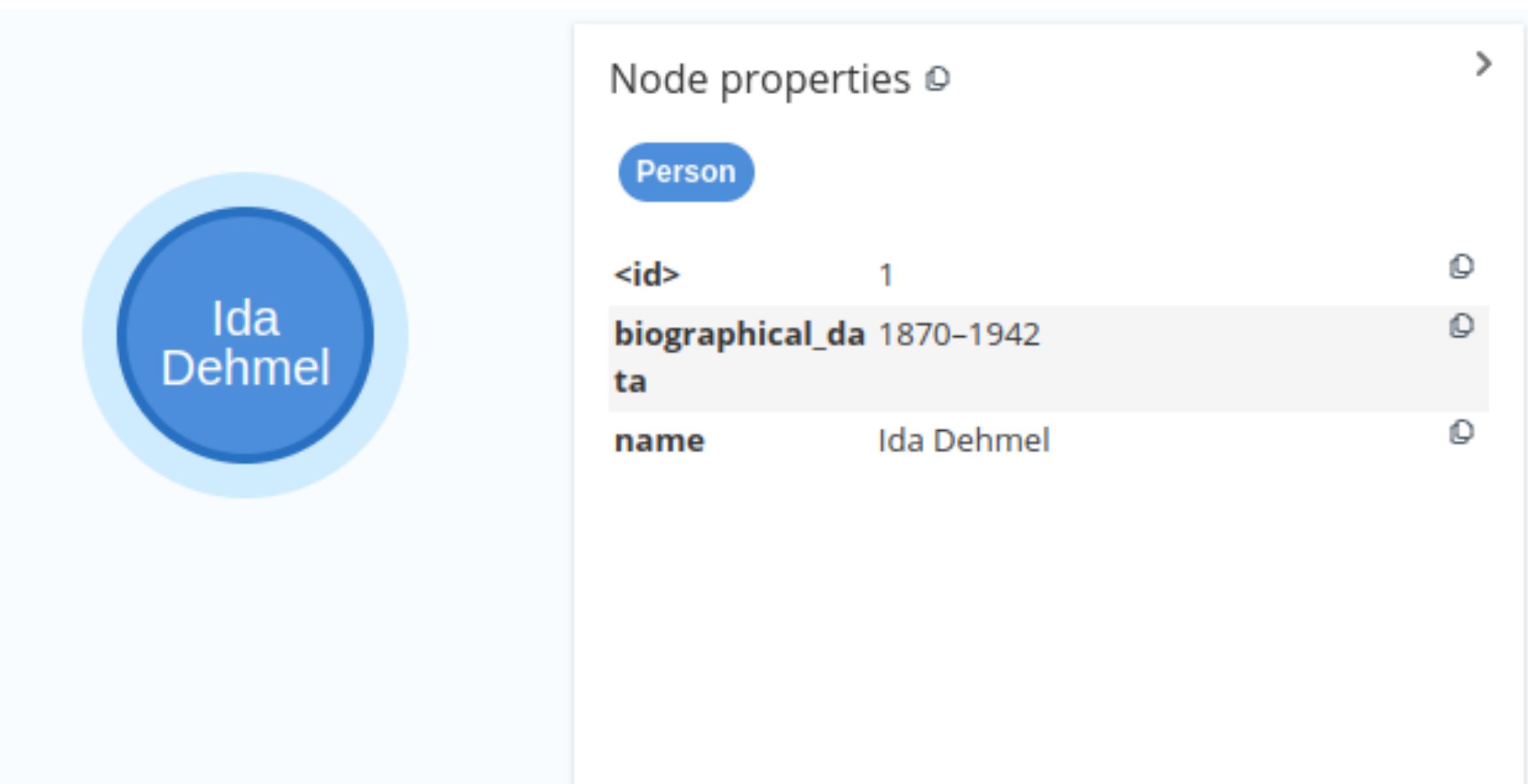
The main area displays the result of the query: "Deleted 1 node, completed after 1 ms."

Eigenschaft ergänzen:

```
MATCH (n:Person {name:'Ida Dehmel'})
```

```
SET n.biographical_data = '1870-1942'
```

```
RETURN n;
```



Nach spezifischen Eigenschaften abfragen:

```
MATCH (n:Person)  
WHERE n.name = 'Ida Dehmel'  
RETURN n;
```



The screenshot shows the Neo4j browser interface. On the left, there is a vertical toolbar with four options: 'Graph' (selected), 'Table', 'Text', and 'Code'. In the main area, a query is entered into the top text input field:

```
neo4j$ MATCH (n:Person) WHERE n.name='Ida Dehmel' RETURN n;
```

Below the query, the results are displayed in a blue circular node. The node contains the text 'Ida Dehmel'.

## Weiteren Knotentyp erstellen

### Knotentyp: Brief

```
neo4j$ CREATE (n:Brief {Datum:'09.05.1907', ID:'HANSb29366'})
```

Added 1 label, created 1 node, set 2 properties, completed after 7 ms.

Table

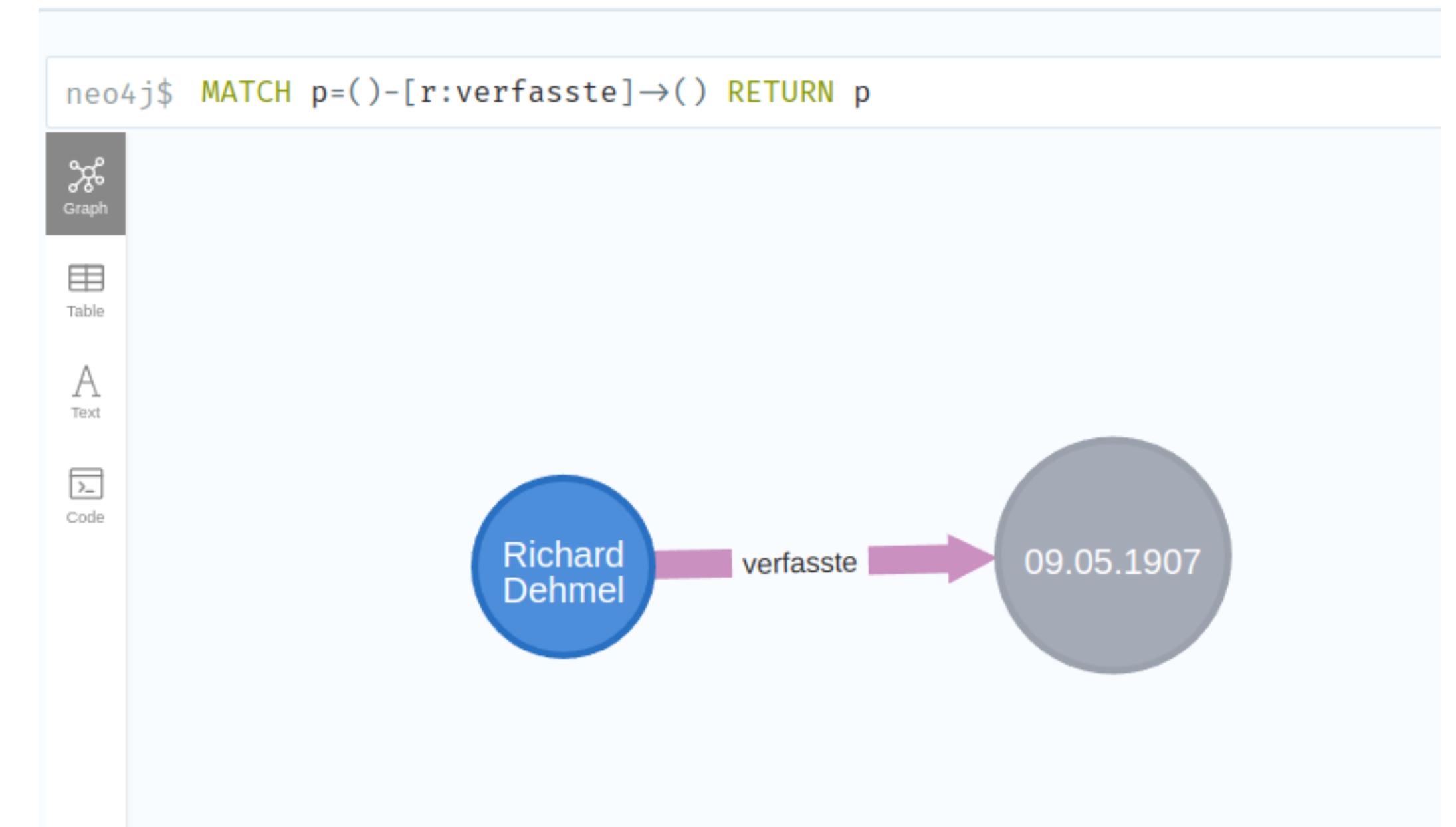
Code

Kante / Relation zwischen zwei Knoten erstellen:

MATCH (n:Person), (b:Brief)

WHERE n.name='Richard Dehmel'  
AND b.ID='HANSb29366'

CREATE (n)-[:verfasste]->(b)



Knoten löschen (Option C):

MATCH (n:Brief)

WHERE ID(n)=0

DETACH DELETE n

```
neo4j$ MATCH (n:Brief) WHERE ID(n)=0 DETACH DELETE n
```



Table



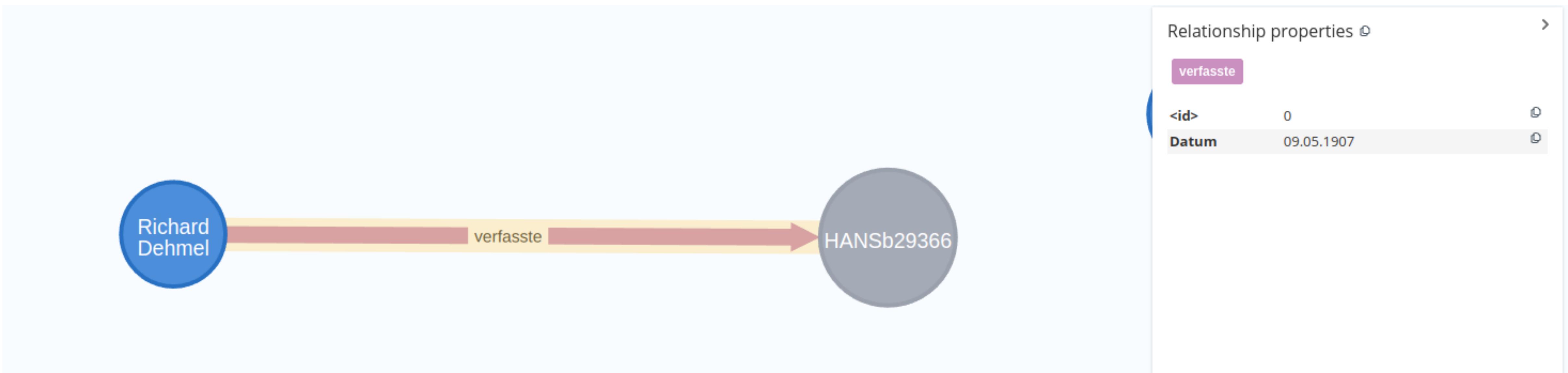
Code

Deleted 1 node, deleted 1 relationship, completed after 4 ms.

Kante mit Eigenschaften:

```
MATCH (n:Person {name:'Richard Dehmel'}), (b:Brief)
```

```
CREATE (n)-[:verfasste {Datum:'09.05.1907"}]->(b)
```

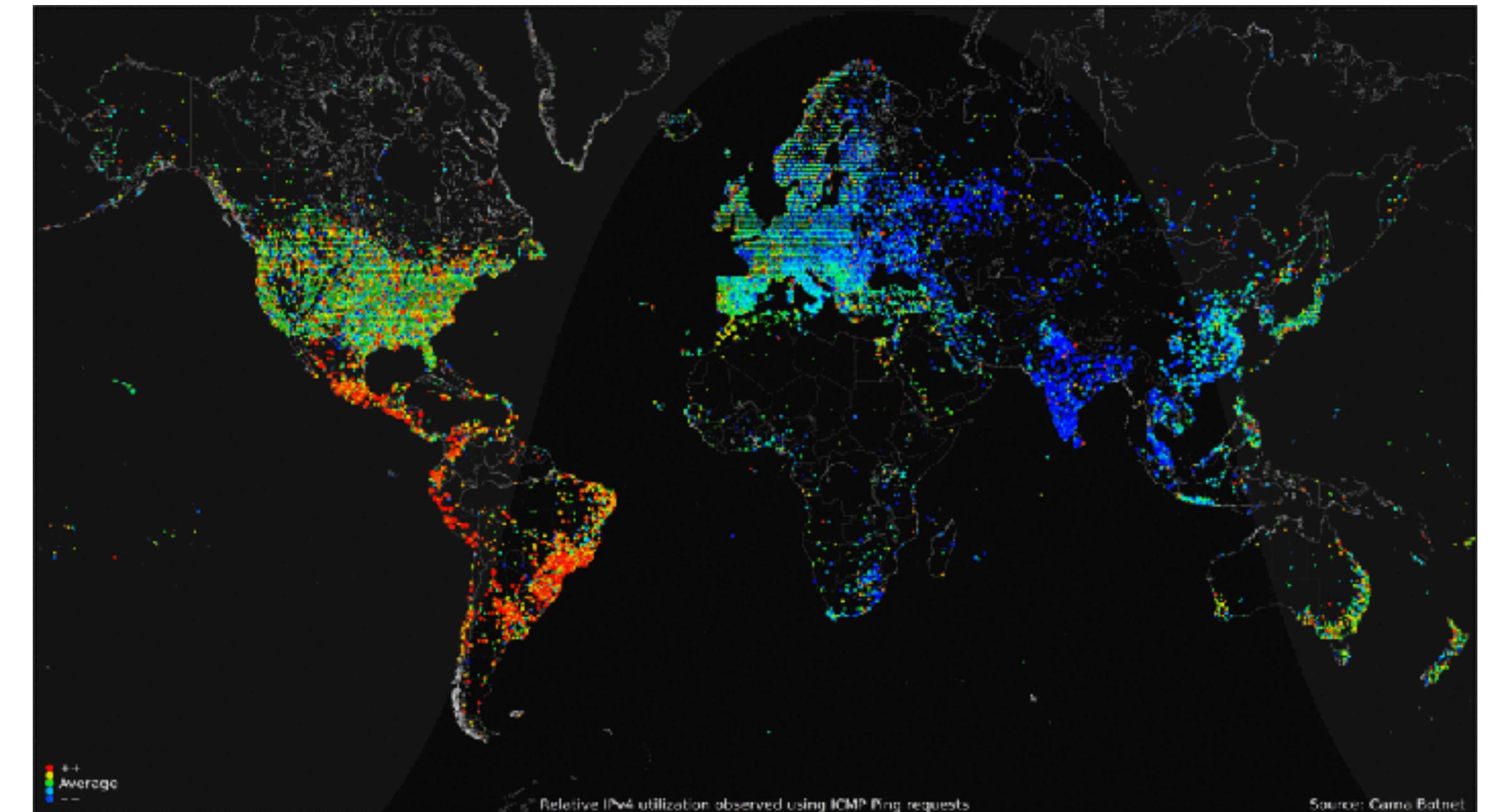


# Die Welt ist ein Graph

## Übung 2

Überdenkt und überarbeitet euer Netzwerk:

- Welche Relationen sollen wie abgebildet werden?
- Welche Entitäten/Knoten benötigen welche Eigenschaften?
- Was wären mögliche Abfragen?



World map of 24 hour relative average utilization of IPv4 addresses observed using ICMP ping requests

<https://om.co/gigaom/check-out-this-visual-map-that-shows-24-hours-of-internet-usage-around-the-world/>

A large hot air balloon is shown from a low angle, filling most of the frame. The balloon's envelope is a vibrant rainbow of colors, transitioning from red at the bottom left to orange, yellow, green, and blue at the top right. The surface has a fine grid pattern of black lines. The basket is visible at the very bottom center. The background is a clear, pale blue sky.

PAUSE

# Datenimport mit csv-Dateien



# CSV-Tabellen

## Vorbereitung der Tabellen

HANS_ID	Author	Author_ID	Recipient	Recipient_ID	Production_Place
HANSb336471	Liliencron, Detlev von	gnd_118572954	Dehmel, Richard	gnd_118679236	
HANSb317396	Wolfthorn, Julie	gnd_1012920224	Dehmel, Ida	gnd_118679228	
HANSb22737	Behrens, Peter	gnd_118508466	Dehmel, Richard	gnd_118679236	Neubabelsberg
HANSb336227	Dehmel, Ida Dehmel, Richard	gnd_118679228 gnd_118679236	Liliencron, Detlev von	gnd_118572954	Hamburg-Blankenese
HANSb315297	Wolfthorn, Julie	gnd_1012920224	Dehmel, Ida	gnd_118679228	Berlin
HANSb336015	Dehmel, Richard	gnd_118679236	Zweig, Stefan	gnd_118637479	
HANSb315263	Wolfthorn, Julie	gnd_1012920224	Dehmel, Ida	gnd_118679228	Berlin
HANSb17272	Liliencron, Detlev von	gnd_118572954	Dehmel, Richard	gnd_118679236	
HANSb336485	Liliencron, Detlev von	gnd_118572954	Dehmel, Richard	gnd_118679236	Hamburg-Rahlstedt
HANSb312845	Dehmel, Richard	gnd_118679236	Behrens, Peter	gnd_118508466	Hamburg-Blankenese
HANSb336482	Liliencron, Detlev von	gnd_118572954	Dehmel, Richard	gnd_118679236	Hamburg-Rahlstedt
HANSb17275	Liliencron, Detlev von	gnd_118572954	Dehmel, Richard	gnd_118679236	Hamburg-Rahlstedt
HANSb315264	Wolfthorn, Julie	gnd_1012920224	Dehmel, Ida	gnd_118679228	Berlin
HANSig398	Behrens, Peter Dehmel, Richard	gnd_118508466 gnd_118679236	Dehmel, Richard	gnd_118679236	Darmstadt
HANSb17423	Liliencron, Detlev von	gnd_118572954	Dehmel, Richard	gnd_118679236	Hamburg-Rahlstedt
HANSb336012	Zweig, Stefan	gnd_118637479	Dehmel, Richard	gnd_118679236	Rangun
HANSb315290	Wolfthorn, Julie	gnd_1012920224	Dehmel, Ida	gnd_118679228	Berlin
HANSb336476	Liliencron, Detlev von	gnd_118572954	Dehmel, Richard	gnd_118679236	Hamburg-Rahlstedt
HANSb312842	Dehmel, Richard	gnd_118679236	Behrens, Peter	gnd_118508466	Hamburg-Blankenese
HANSb336449	Liliencron, Detlev von	gnd_118572954	Dehmel, Richard	gnd_118679236	Hamburg-Rahlstedt
HANSb315252	Wolfthorn, Julie	gnd_1012920224	Dehmel, Ida	gnd_118679228	Colmar
HANSb337102	Liliencron, Detlev von	gnd_118572954	Dehmel, Richard	gnd_118679236	
HANSb336024	Zweig, Stefan	gnd_118637479	Dehmel, Richard	gnd_118679236	Wien
HANSb336216	Dehmel, Richard	gnd_118679236	Liliencron, Detlev von	gnd_118572954	Hamburg-Blankenese
HANSb22739	Behrens, Peter Bachmann, Alf	gnd_118508466 gnd_119107457	Dehmel, Richard	gnd_118679236	Düsseldorf

# CSV

## Github-Import

- Github-Daten via "Raw"-Button aufrufen
- Link kopieren und in neo4j-Prompt einsetzen

```
LOAD CSV FROM "https://raw.githubusercontent.com/FranziskaDigital/Dehmel_Test/main/Personen_GND.csv?token=GHSAT0AAAAAAB7JKATI2IBTNCMVV7563MW4Y73QWHQ" AS row  
fieldterminator ';'  
| CREATE (p:Person {name:row[0], gnd:row[1]})
```

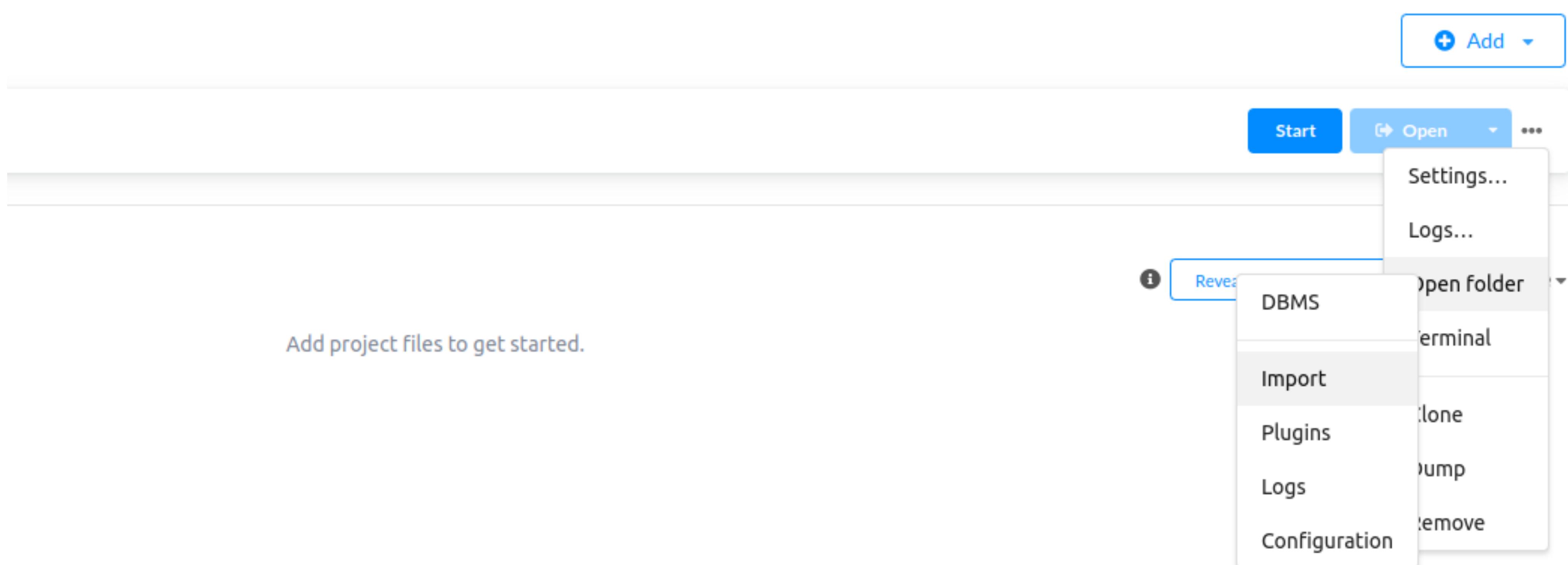
Added 88 labels, created 88 nodes, set 176 properties, completed after 276 ms.

# CSV

## Import von lokalem Speicherort

Datei in neo4j-Import-Ordner legen

DB starten



# CSV-Import von lokalem Speicherort

```
LOAD CSV FROM "file:///Briefe.csv" AS row fieldterminator ";"
```

```
CREATE (:Brief {ID:row[0]}, Autor_in:row[1], Empfaenger_in:row[3])
```

```
neo4j$ LOAD CSV FROM "file:///Briefe.csv" AS row fieldterminator ";" CREATE (:Brief {ID:row[0]}, Autor_in:row[1], Empfaenger_in:row[3])
```

Added 953 labels, created 953 nodes, set 2858 properties, completed after 117 ms.

Table

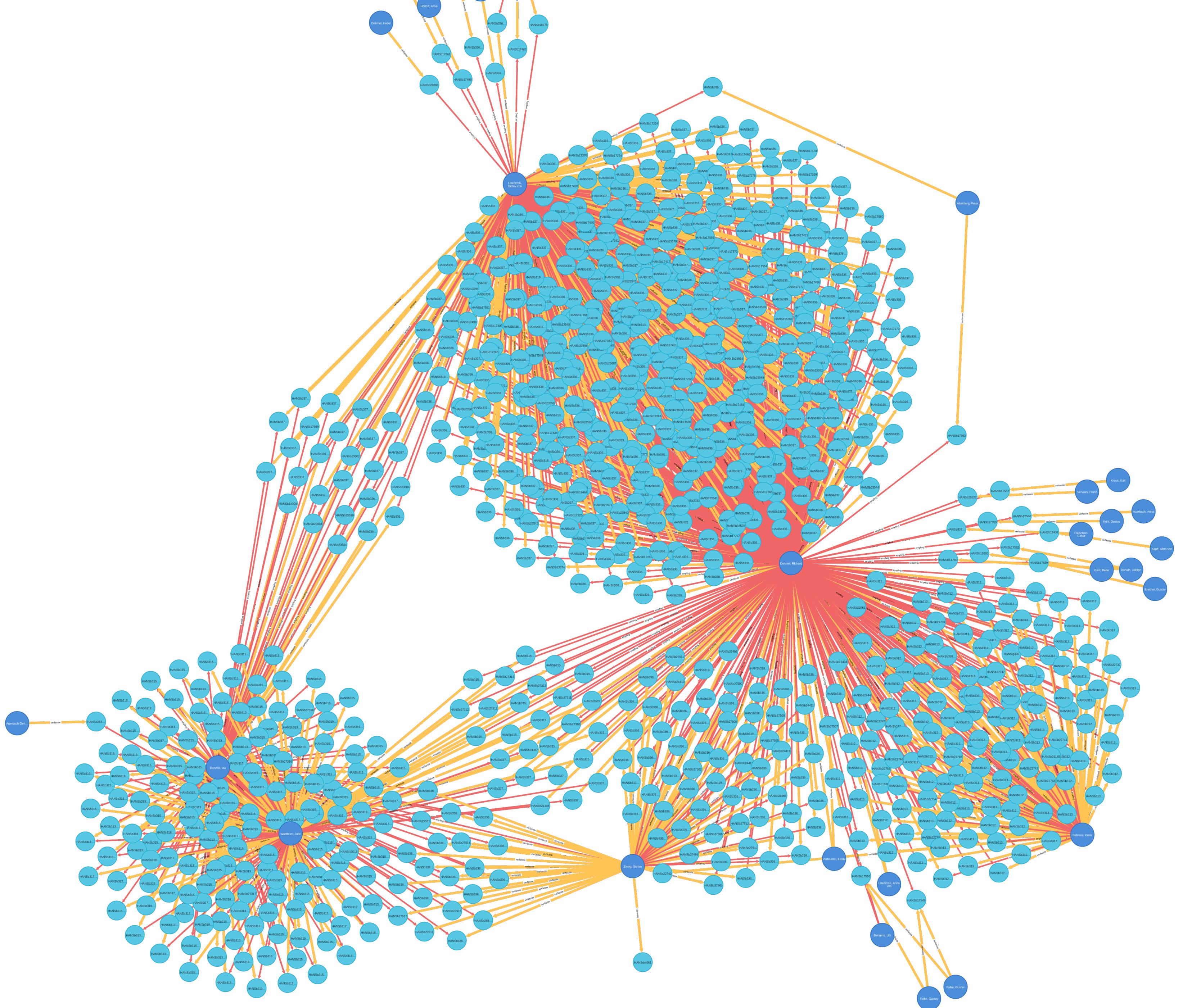
Code

# **Relationen erstellen nach CSV-Import**

```
MATCH (b:Briefe), (p:Person)
```

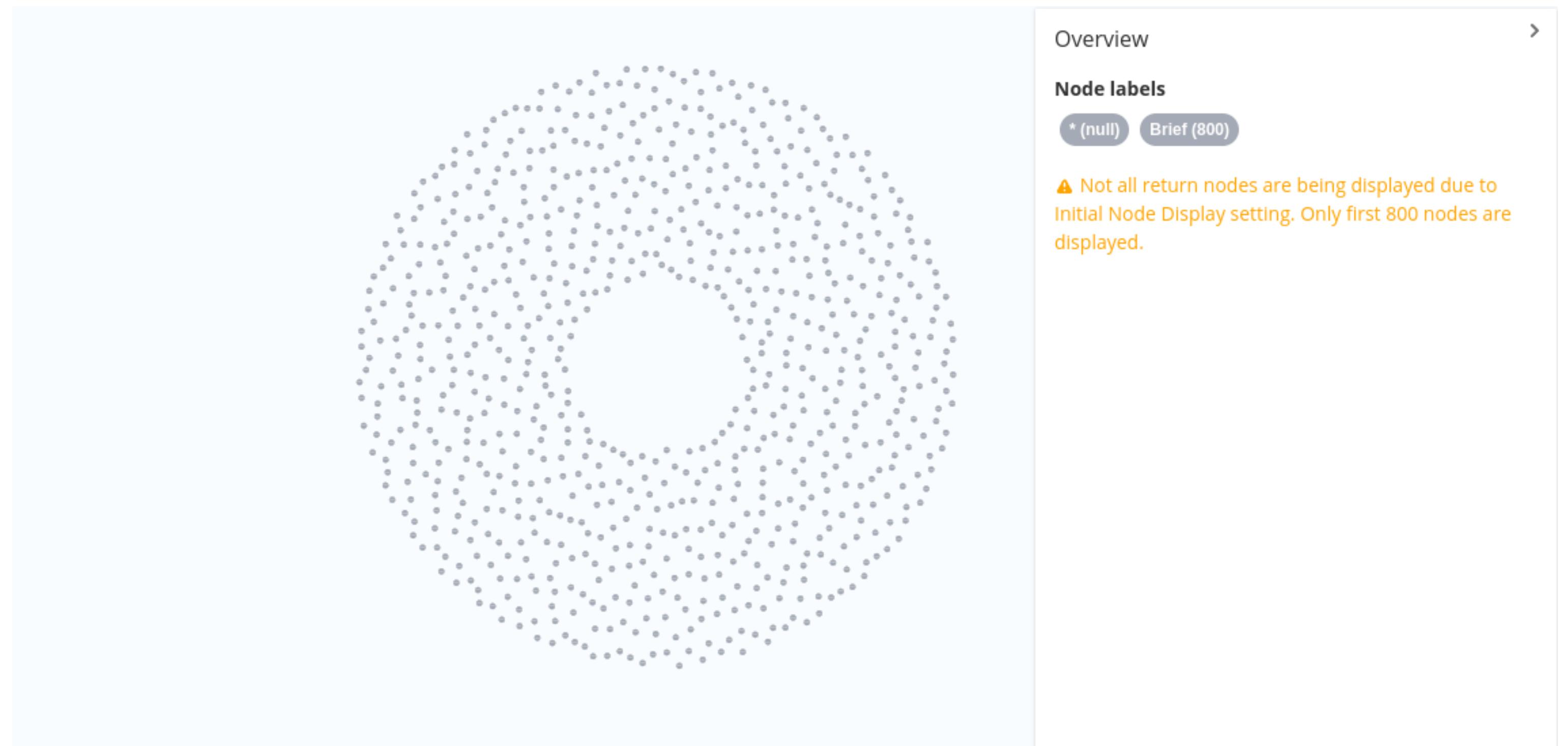
```
WHERE b.Autor_in = p.Name
```

```
CREATE (p)-[:verfasste]->(b)
```



# Abfragen nach Knotentyp / Label

```
MATCH (n:Brief) RETURN n
```



# Abfragen nach Eigenschaft

```
1 MATCH (n:Brief)
2 WHERE n.Empfaenger_in='Dehmel, Richard'
3 RETURN n
```

The screenshot shows a Neo4j browser interface. On the left, there's a sidebar with four tabs: Graph (selected), Table, Text, and Code. The main area displays a large, dense network graph consisting of numerous small grey dots representing nodes, forming a roughly circular cluster. In the top right corner of the main area, there are icons for running the query, saving it, and closing the window. To the right of the graph, there's an 'Overview' section. It shows 'Node labels' with two entries: '\*' (558) and 'Brief' (558). Below that, it says 'Displaying 558 nodes, 0 relationships.' At the bottom right of the main area, there are three small icons for zooming in, zooming out, and switching between node and relationship views.

# **Abfragen**

## **nach Relation**

```
MATCH p=()-[r:verfasste]->() RETURN p
```

# Abfragen

## nach Häufigkeit

```
MATCH (a:Person)-[v:verfasste]->(b:Briefe)
WITH a, count(v) as Relation, collect (v) as verfasste WHERE Relation >1
RETURN a as Person, Relation as Anzahl ORDER BY Anzahl DESC
```

```
1 MATCH (a:Person)-[v:verfasste]→(b:Briefe) WITH a, count(v) as Relation, collect(v) as verfasste WHERE Relation >1  
2 RETURN a as Person, Relation as Anzahl ORDER BY Anzahl DESC
```

Graph

Table

A  
Text

Code

"Person"	"Anzahl"
{"GND":"gnd_118572954", "Name":"Liliencron, Detlev von"}	397
{"GND":"gnd_118679236", "Name":"Dehmel, Richard"}	171
{"GND":"gnd_1012920224", "Name":"Wolfthorn, Julie"}	129
{"GND":"gnd_118508466", "Name":"Behrens, Peter"}	105
{"GND":"gnd_118637479", "Name":"Zweig, Stefan"}	69
{"GND":"gnd_118679228", "Name":"Dehmel, Ida"}	55
{"GND":"gnd_118502255", "Name":"Altenberg, Peter"}	2
{"GND":"gnd_118685929", "Name":"Falke, Gustav"}	2
{"GND":"gnd_11947042X", "Name":"Donath, Adolph"}	2
{"GND":"gnd_118685929", "Name":"Falke, Gustav"}	2

MAX COLUMN WIDTH:

# Abfragen

## Triangle Count

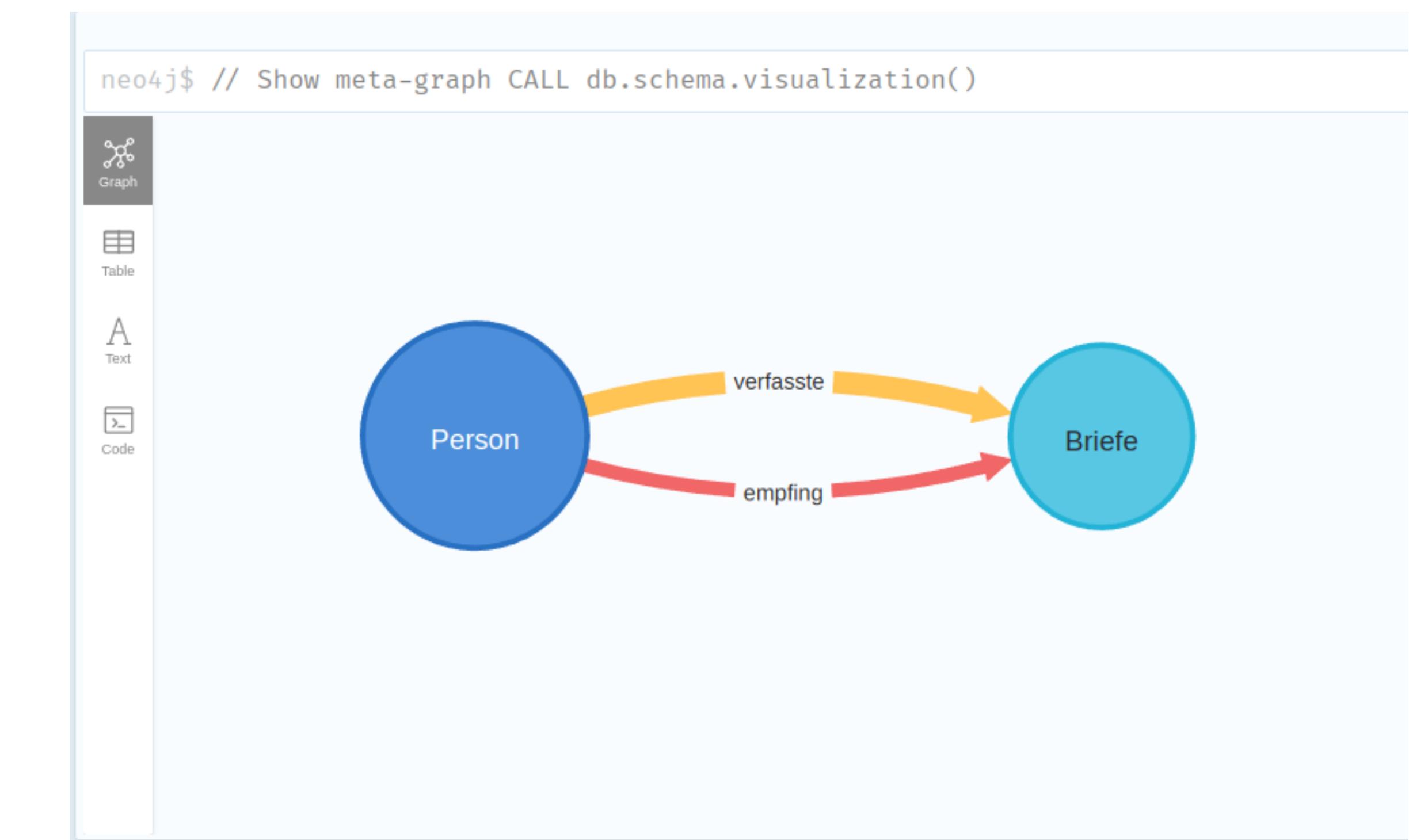
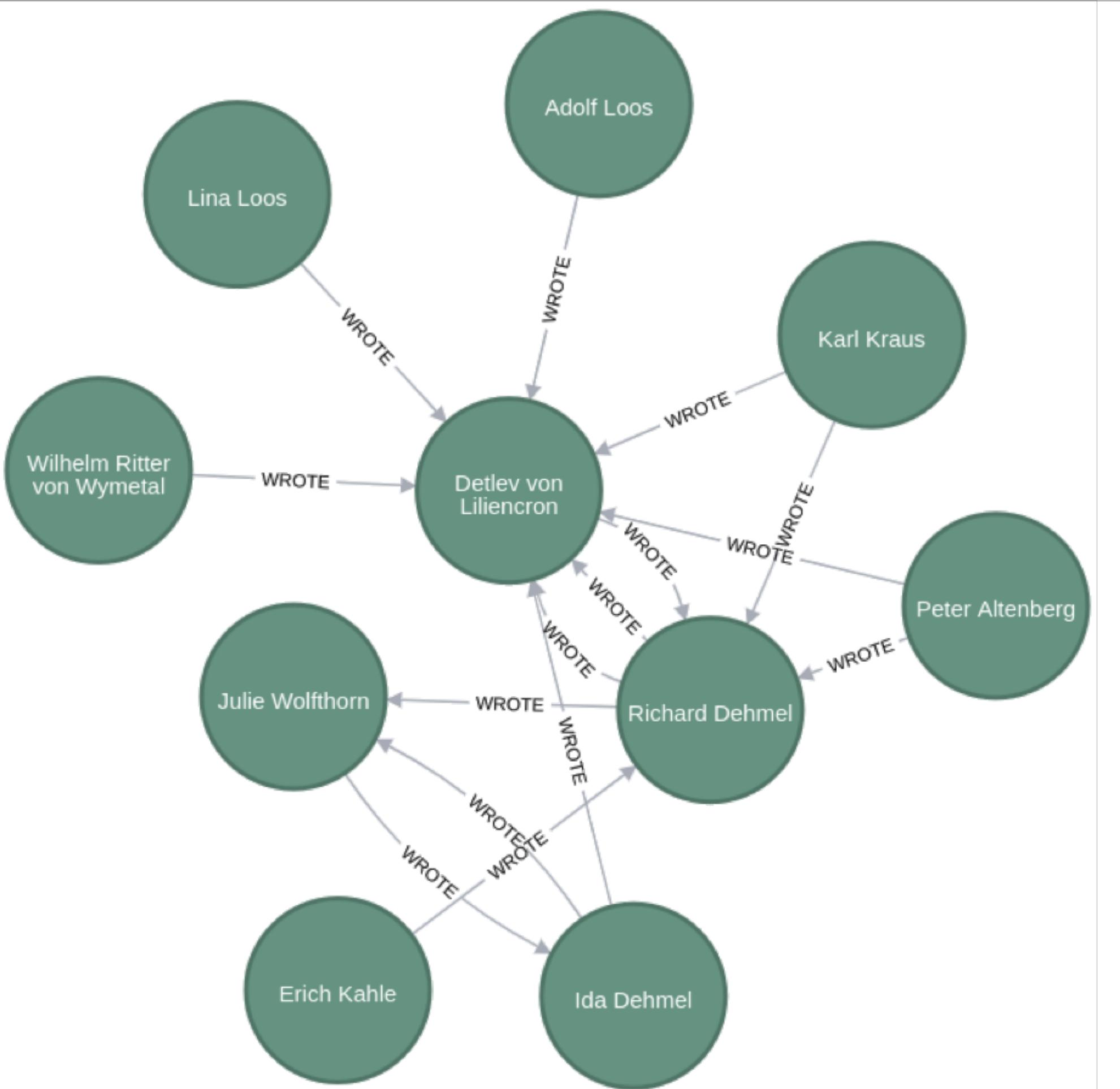
```
1 CALL gds.graph.project(  
2   'Dehmel_Project',  
3   'Person',  
4   {  
5     verfasste: {  
6       orientation: 'UNDIRECTED'  
7     }  
8   }  
9 )
```

Table

"nodeProjection"	"relationshipProjection"	"graphName"	"nodeCount"	"relationshipCount"	"projectMillis"
{"Person":{"label":"Person","properties":{}}}	{"verfasste":{"orientation":"UNDIRECTED","indexInverse":false,"aggregation":"DEFAULT","type":"verfasste","properties":{}}}	"Dehmel_Project"	88	0	595

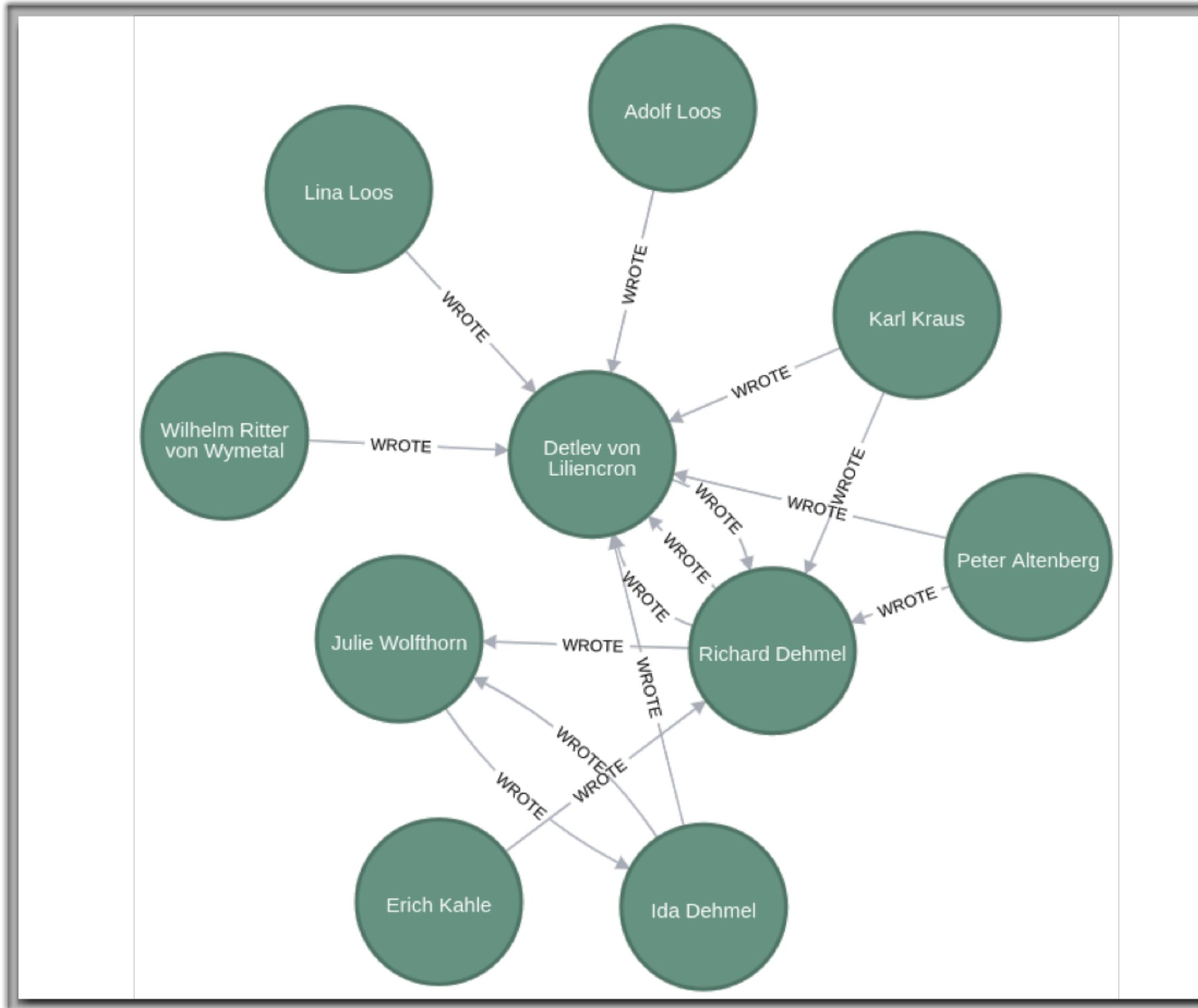
A  
Text

Code



# Abfragen

## Triangle Count



nodeA	nodeB	nodeC
Detlev von Liliencron	Peter Altenberg	Richard Dehmel
Detlev von Liliencron	Karl Kraus	Richard Dehmel

Name	triangleCount
Detlev von Liliencron	2
Richard Dehmel	2
Peter Altenberg	1
Karl Kraus	1
Julie Wolfthorn	0
Adolf Loos	0
Lina Loos	0
Wilhelm Ritter von Wymetal	0
Erich Kahle	0
Ida Dehmel	0

Weitere Infos dazu:

<https://neo4j.com/docs/graph-data-science/current/algorithms/triangle-count/#algorithms-triangle-count-examples-stream>

# Abfragen

## Clustering Coefficient

The screenshot shows a Neo4j browser window with the following details:

- Code Tab:** Contains the Cypher query:

```
1 CALL gds.localClusteringCoefficient.stream('Dehmel_Project')
2 YIELD nodeId, localClusteringCoefficient
3 RETURN gds.util.asNode(nodeId).Name AS Name, localClusteringCoefficient
4 ORDER BY localClusteringCoefficient DESC
```
- Table:** A result table with two columns: "Name" and "localClusteringCoefficient". The table contains 7 rows, indexed 1 to 7. All entries have a local clustering coefficient of 0.0.
- Text:** Below the table, it says "Started streaming 88 records after 1 ms and completed after 9 ms."

	Name	localClusteringCoefficient
1	"Altenberg, Peter"	0.0
2	"Dehmel, Ida"	0.0
3	"Dehmel, Richard"	0.0
4	"Kraus, Karl"	0.0
5	"Loos, Adolf"	0.0
6	"Loos, Lina"	0.0
7		

# Abfragen

## Clustering Coefficient



Name	localClusteringCoefficient
Peter Altenberg	1.0
Karl Kraus	1.0
Richard Dehmel	0.2
Detlev von Liliencron	0.09523809523809523
Julie Wolfthorn	0.0
Adolf Loos	0.0
Lina Loos	0.0
Wilhelm Ritter von Wymetal	0.0
Erich Kahle	0.0
Ida Dehmel	0.0

# **Erweiterungen, Ergänzungen und Ausblick**



# Mögliche Erweiterungen

## Anpassungen am Datenmodell

- Daten und Modell immer wieder abgleichen und überprüfen
- Möglichkeit der Erweiterung überprüfen
- sinnvolle Erweiterungen können sein: Beziehungen zwischen Personen, Ortsangaben, Zugehörigkeiten zu Institutionen u.ä.
- Eigenschaften können zu Knoten ergänzt werden, um Abfragen zu erweitern oder zu präzisieren