Implementation by Hossein

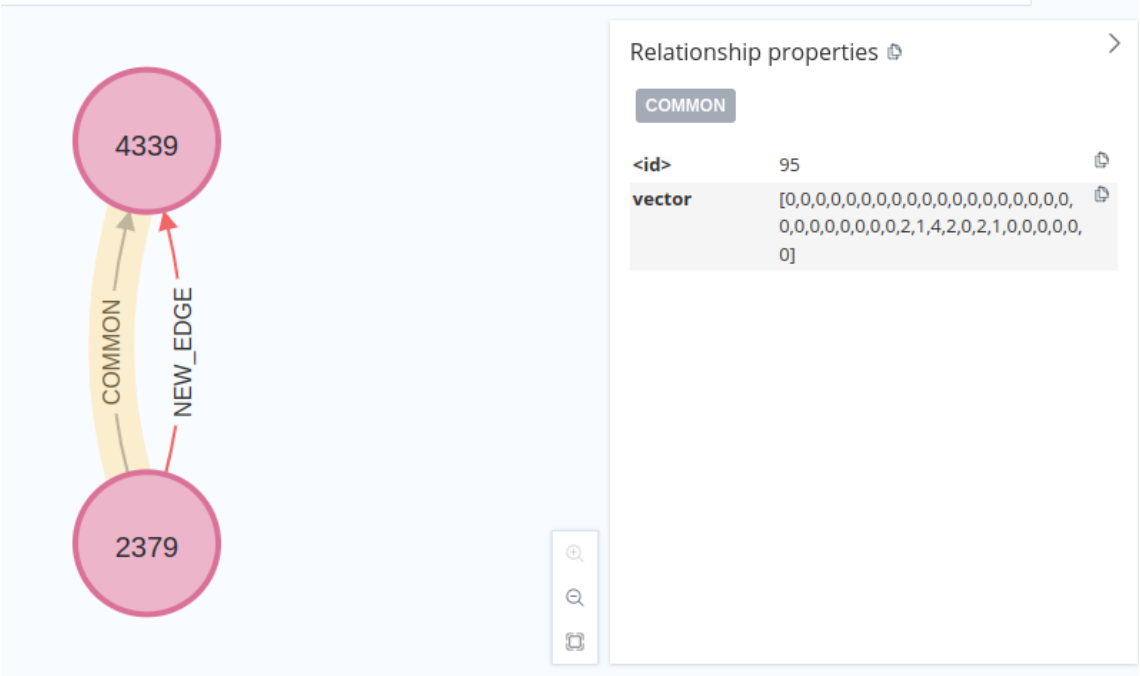
Objective 1: Creating a graph where Musicians are nodes, each having a musician ID, and edges between two musicians are discrete (40,1) vectors. Each entry corresponds to the number of sessions those two musicians had in that given year. Given that years start from 1949 to 1969 and span a 40-year interval, this vector neatly models the number of sessions. We can therefore decide the number of common sessions of any two musicians for any subset of given years by applying a binary mask.

Phase 0: Proof of concept

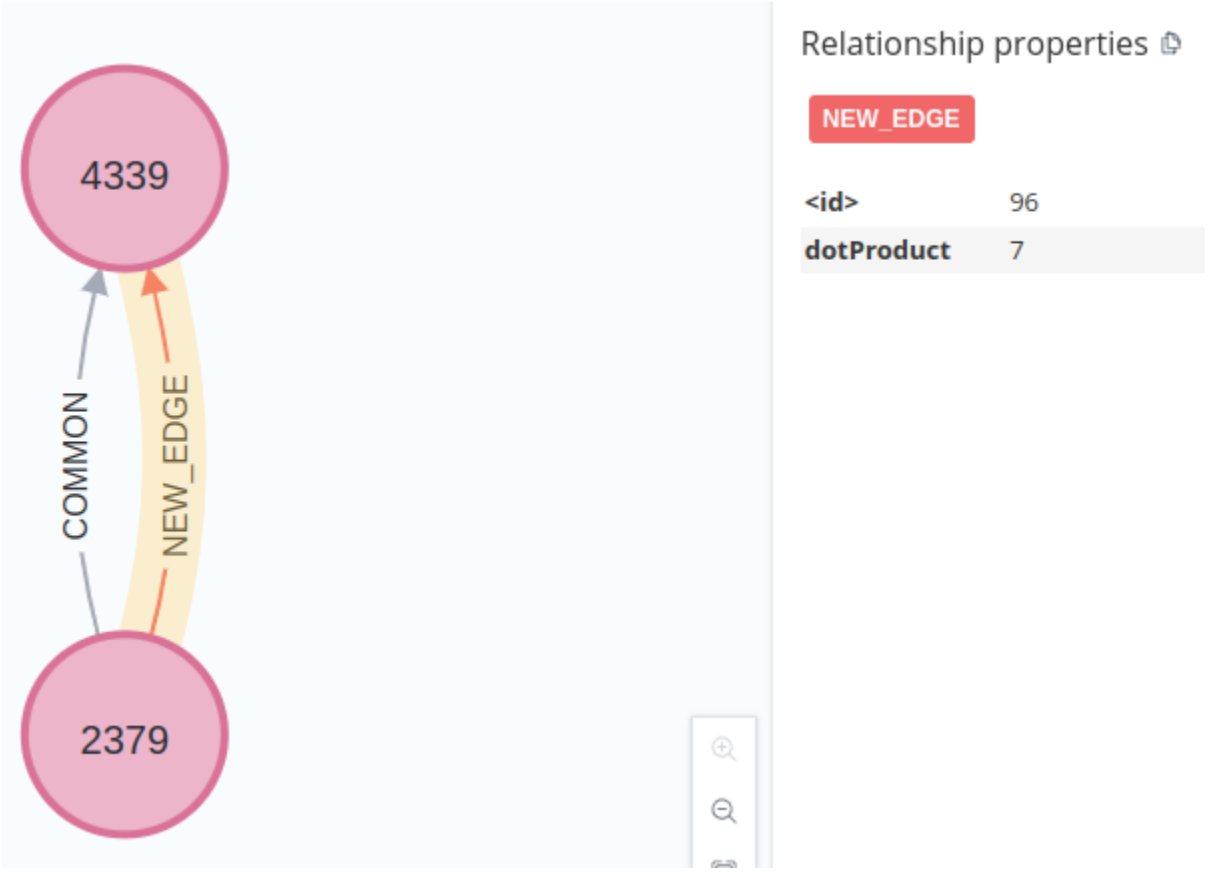
For two musicians of interest, we show that our proposed approach work. Take musicians 2379 and 4339. This is their statistics regarding common sessions:

	123 id_musician	123 id_musician	123 session_year	123 occurrence
1	2,379	4,339	1,956	3
2	2,379	4,339	1,957	1
3	2,379	4,339	1,958	4
4	2,379	4,339	1,959	2
5	2,379	4,339	1,961	2
6	2,379	4,339	1,962	1

COMMON edge refers to the raw data regarding session information. Here it correctly shows the session information of the two musicians of interest. The nodes and COMMON edges will be build only once.



Now for years, say 1957 to 1959, after giving these two years to a python script and obtaining the mask and the query, we can create the NEW_EDGE. As seen, the value is correctly 7 = 1 + 4 + 2



Conclusion: This approach seems feasible and worth trying. Given the size of the dataset and features, it seems queries could be run pretty fast.

Phase 1: Running on large scale
Due to compute limitations, the next phase shall be run on Valhalla

Step 1: Upgrade Neo4J to version 5, get access remotely to the browser and upload the csv file to the import folder (Not completed yet)
Proceed with version 4

Step 2: Build the dataset and save the graph and monitor the build time
Done. Build time is < 5 minutes. Is only incurred once
Original graph created with year vectors. Mask is also successfully created

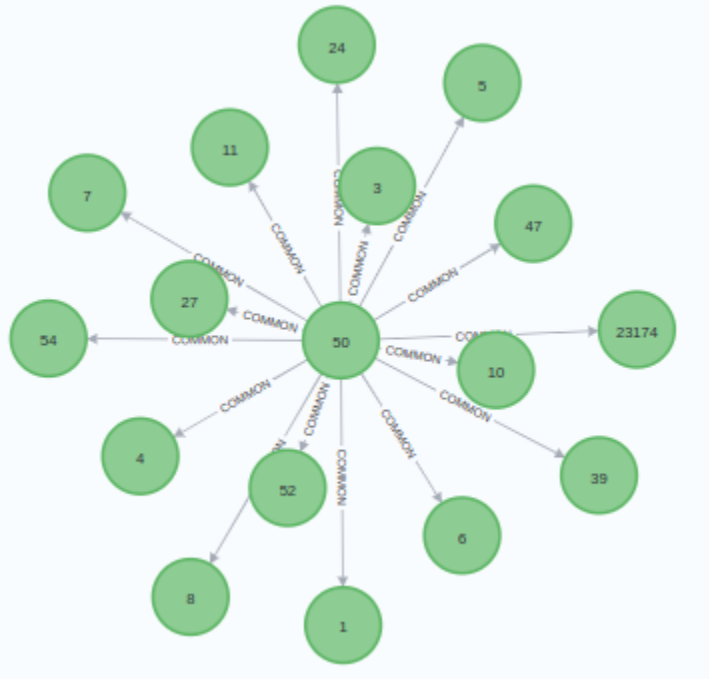
Nodes are not consistent between edges and musicians hehe

Step 3: Run a few queries and assess performance
My implementation now supports this query:
Given a specific musician_id, and given an arbitrary set of years, we can find those musicians with which our specific musician has had a session. Edges in the graph represent the number of sessions for that time period.

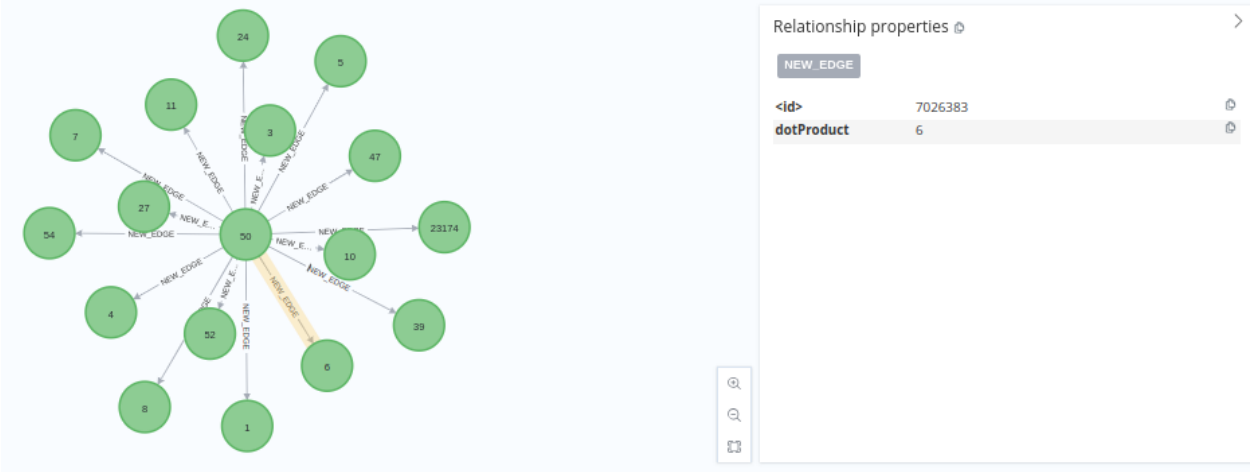
Now let's walk through a sample of probing the musician with musician_id = 50

Bellow is the whole sessions the node with musician_id = 50 has had in his career

Below is the COMMON edges, were for this musician, we create an edge with all of its collaborators. The edge has a discrete vector attribute that mentions how many sessions they had during each year:



Now for instance if we choose 6, we see that 50 has had 6 sessions in total in the selected timeframe (1929-1940):



The above result can easily be verified by checking the database tables, where we see 6 and 50 have had 6 common sessions in their careers:

```
select * from hm31.curated_session_musician_years csmy where csmy.musician_1 = 50 and csmy.musician_2 = 6;
```

curated_session_musician_years 1

```
select * from hm31.curated_session_musician_
```

Enter a SQL expression to filter results (use Ctrl+Space)

	123 musician_1	123 musician_2	123 year
1	50	6	1,938
2	50	6	1,938
3	50	6	1,938
4	50	6	1,939
5	50	6	1,939
6	50	6	1,940

This process can be repeated for any arbitrary node.

Demo can be seen here