Introduction to Database Systems: CS305 Neo4J: building your own graphs

Oliver Bonham-Carter
Hang Zhao

30 November 2023

Databases, Visually

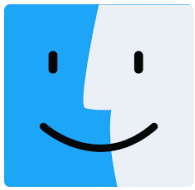
- A visual database system using methods from graph theory to use networks to determine relationships (edges) and discover meaning from connected data-points (nodes). Users are able to interact with the data in a network.

- `https://neo4j.com/`
- Graphgists Projects: `https://neo4j.com/graphgists/`

Getting started with Neo4j in Docker

These files are located in `sandbox/`





Windows

```
build_neo4j_windows.bat
```

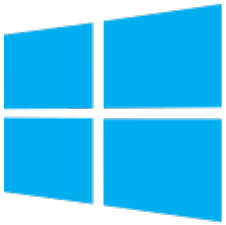
MacOS and Linux

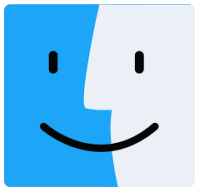
```
sh build_neo4j_macOSAndLinux.sh
```

You can **build** and **start** the container with this script. You will have to manually stop the container, as necessary.

Getting started with Neo4j in Docker

Specific Terminal commands





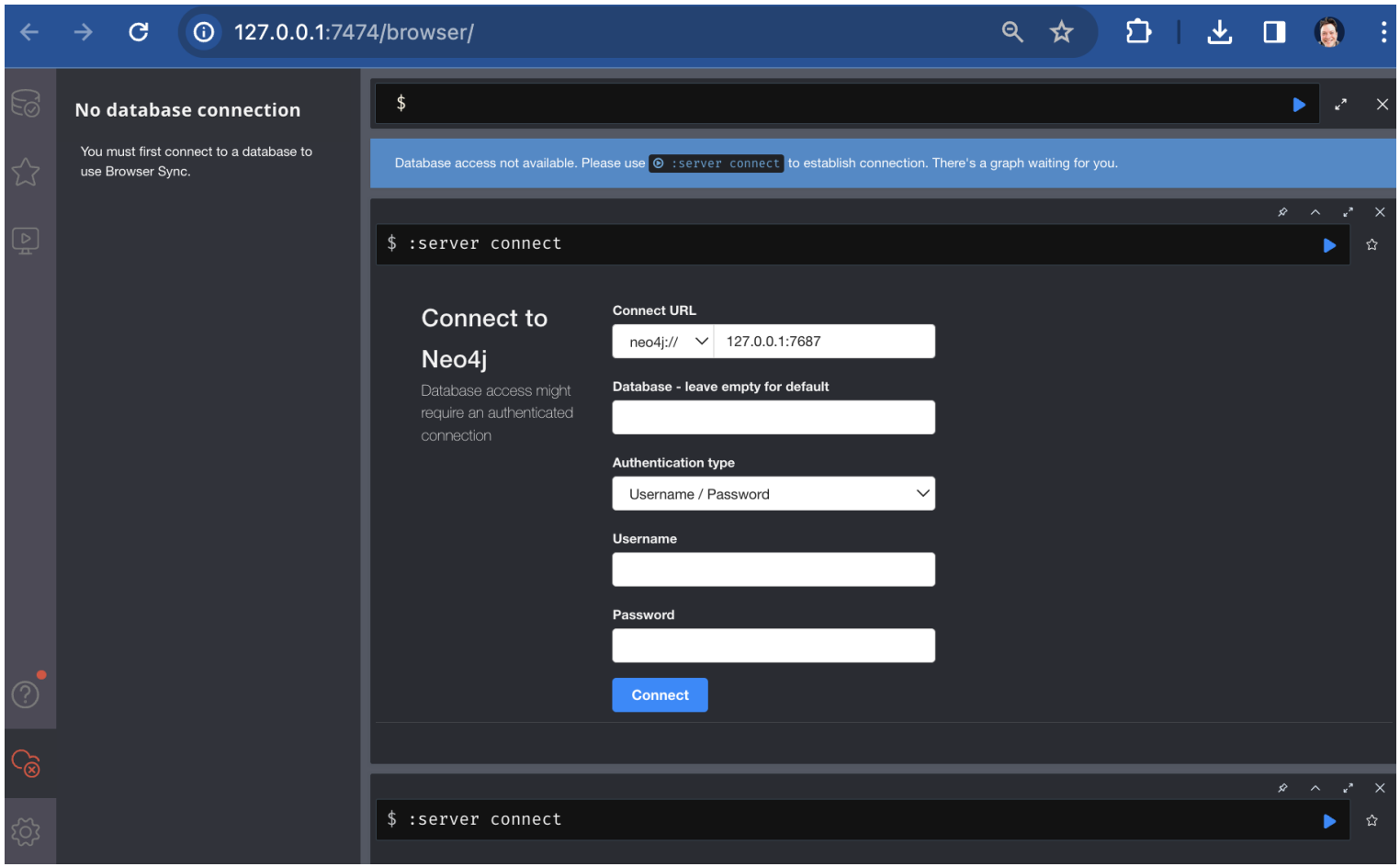
Terminal Command to START Neo4j

```
docker start testneo4j # windows
sudo docker start testneo4j # MacOS and Linux
```

Terminal Command to STOP Neo4j

```
docker stop testneo4j # windows
sudo docker stop testneo4j # MacOS and Linux
```

Login



- Open your browser and head to: `http://127.0.0.1:7474/browser/`

User and Password

Note: The user and password variables are defined in the *build* files we used to create the Docker container.

- Your first login
  - **User**: `neo4j`
  - **Password**: `password`

Parameter in the build file

```
--env NEO4J_AUTH=neo4j/password
```

Add Nodes

File: sandbox/classroomBuild.txt

Destroy all nodes in the graph and erase the graph

```
MATCH (n) DETACH DELETE (n)
```

Add the nodes

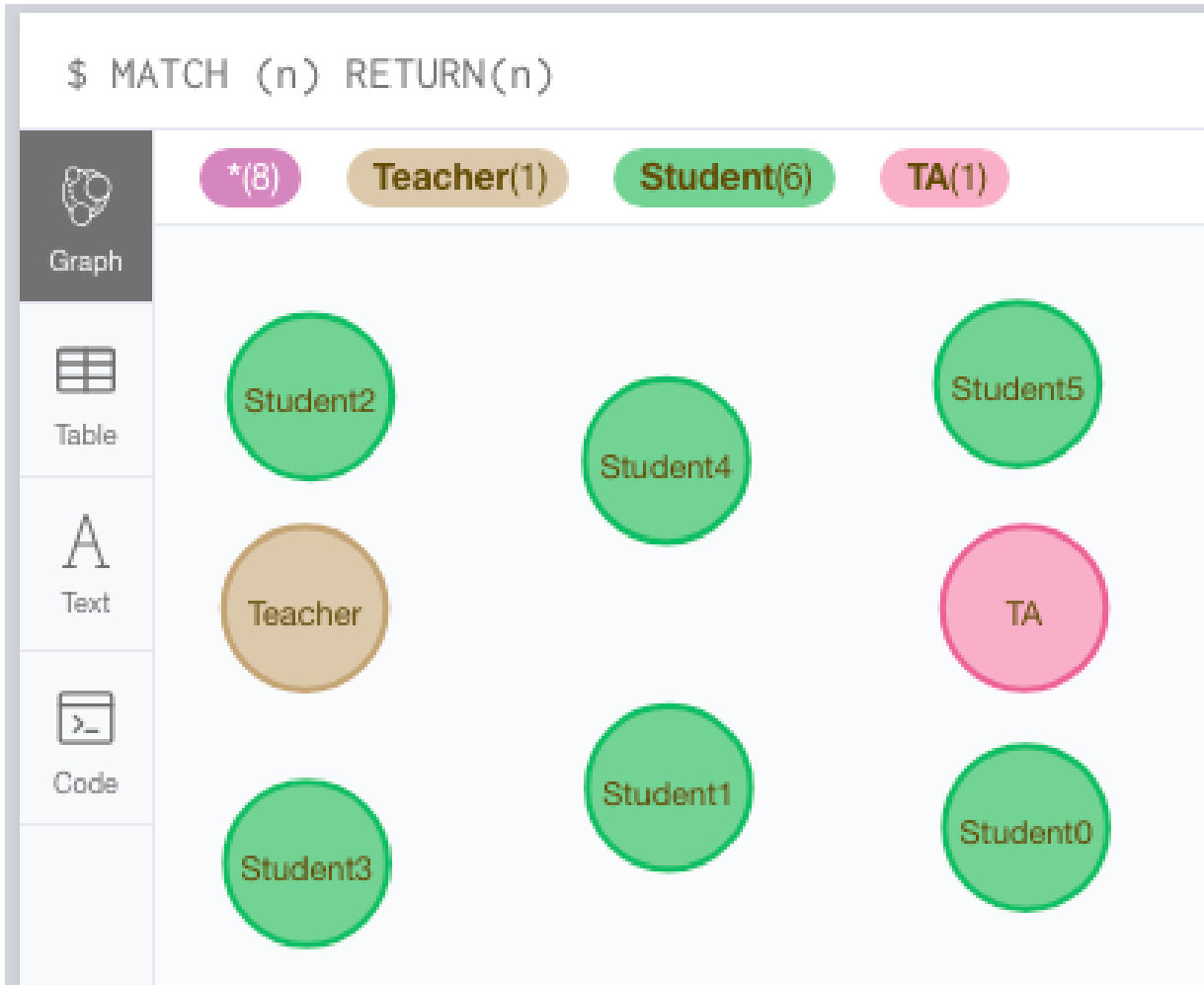
```
CREATE (
   :Teacher {
   name: "Teacher",
   Jackjet: "green",
   Jeans: "blue",
   MarkerCol: "red"}
 )
 FOREACH (r IN range(0,5)|
    CREATE (
      :Student { name:"Student" + r,
      extraUtility: "backpack" + r,
      lastTestScore:tan(rand())*100 })
      )
CREATE (:TA { name: "TA", Machine: "Laptop"})
```

- Adds nodes with some meta data: a Teacher, a TA and five Student

Show the Nodes

Show the unconnected graph

```
MATCH (n) RETURN (n)
```



Add Edges

Add some connectivity to nodes

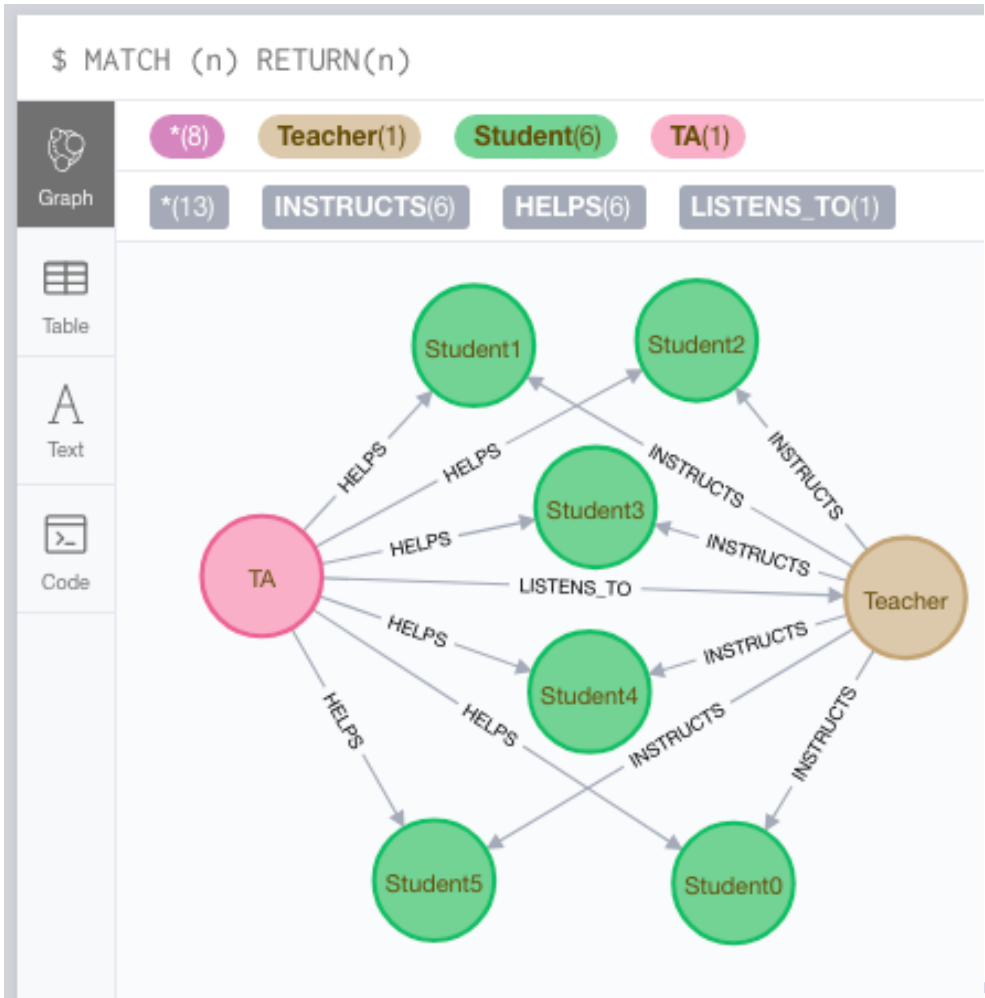
```
MATCH (t:Teacher), (s:Student), (a:TA)
MERGE (t) - [:INSTRUCTS] -> (s) <-[:HELPS] - (a)
MERGE (a) - [:LISTENS_TO] -> (t)
MERGE (t) - [:INSTRUCTS] -> (a) <-[:HELPS] - (s)
```

- The `Teacher` (t) and `Student` (s) nodes are linked by *INSTRUCTS* and an arrow to show direction, $->$
- The `TA` (a) and `Student` (s) nodes are linked by *HELPS* and an arrow to show direction, $<-$
- The `TA` (a) and `TEACHER` (t) nodes are linked by *LISTENS_TO* and an arrow to show direction, $->$
- The `Teacher` (t) and `TA` (a) nodes are linked by *INSTRUCTS* and an arrow to show direction, $->$
- The `Student` (s) and `TA` (a) nodes are linked by *HELPS* and an arrow to show direction, $<-$

Show The Edges

Show the connected graph

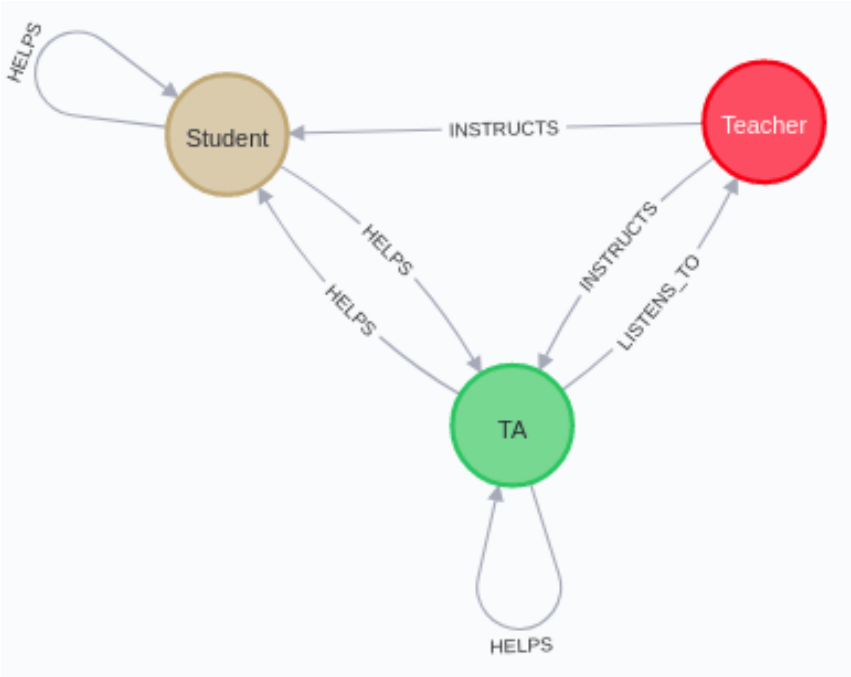
```
MATCH (n) RETURN (n)
```



Schema

Show the schema

```
call db.schema.visualization
```

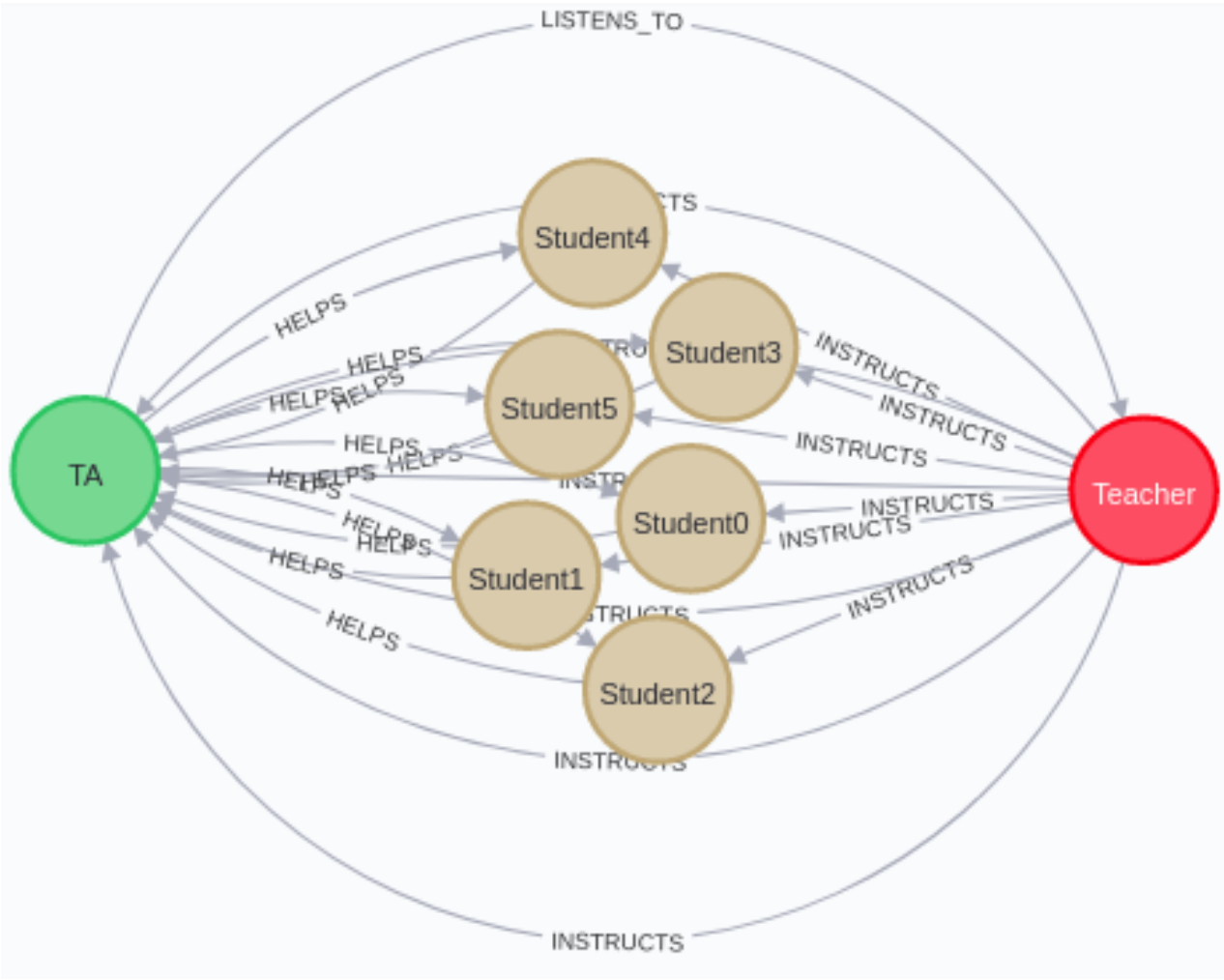


- The `Teacher` Instructs each `Student`
- The `Student` is **Instructed** and **Helped** by `Teacher`
- The `TA` is **Instructed** by `Teacher` and **Listens to** `Teacher`, **Helps** `Student` and self.

Relationship Queries

Who instructs whom?

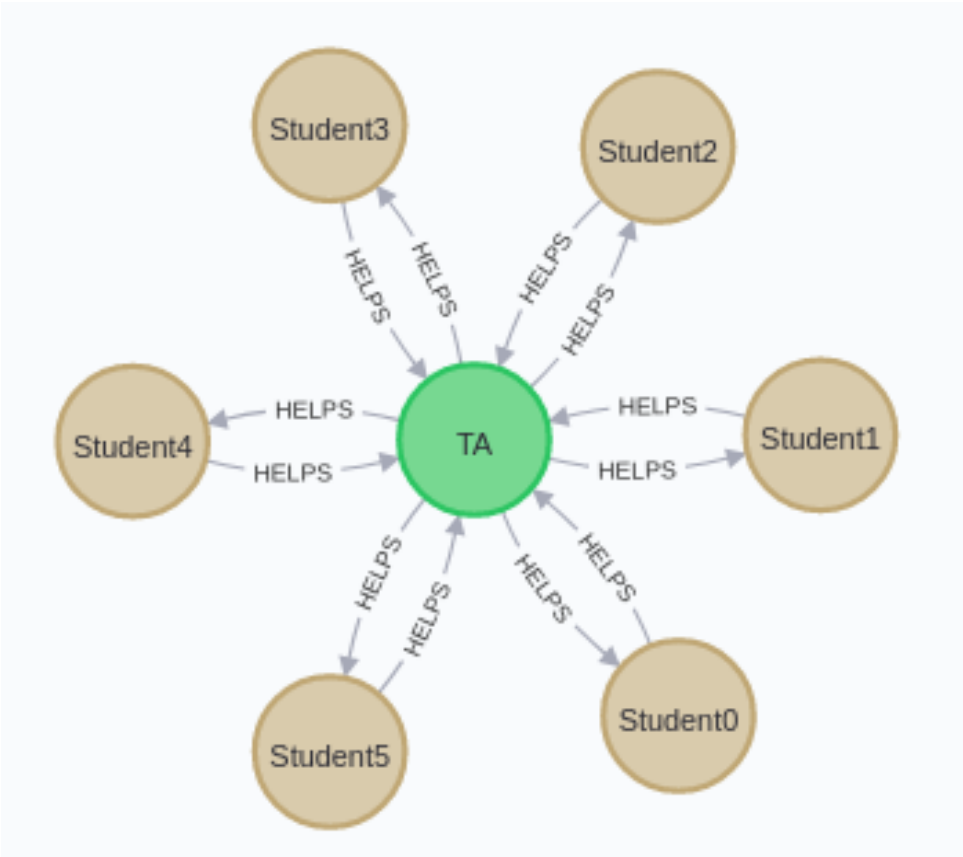
```
MATCH t=()-[s:INSTRUCTS]->() RETURN t
```



Relationship Queries

Who helps whom?

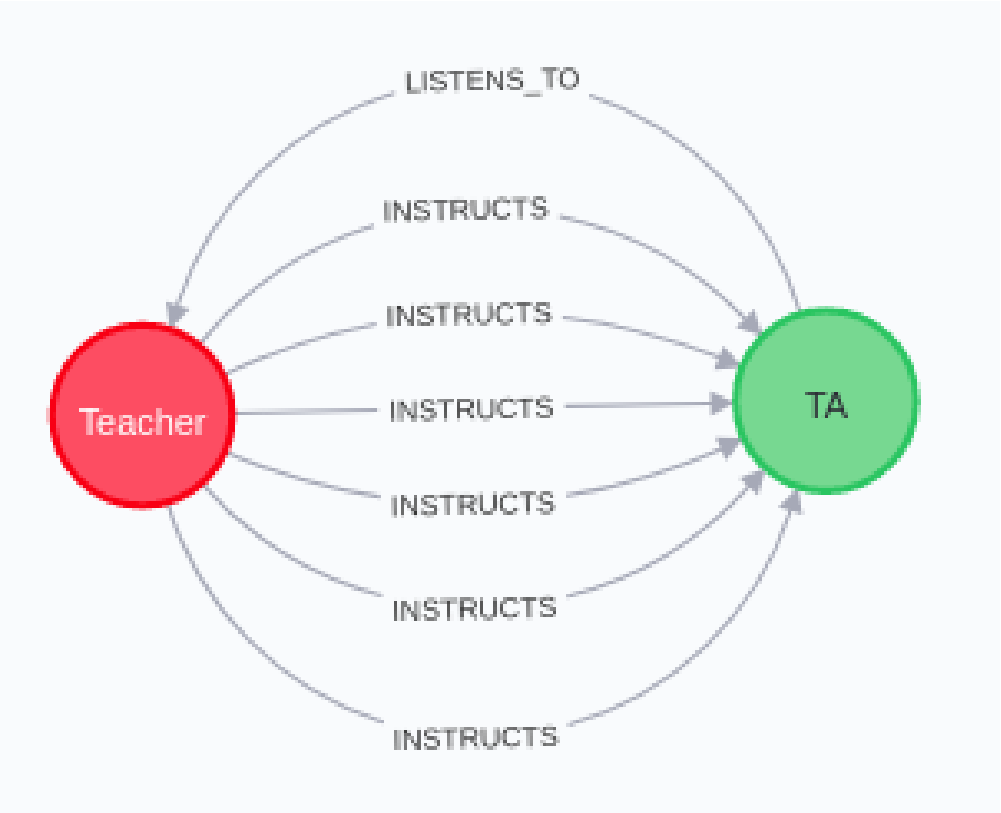
```
MATCH t=()-[s:HELPS]->() RETURN t
```



Relationship Queries

Who listens to whom?

```
MATCH t=()-[s:LISTENS_TO]->() RETURN t
```



Commonly Used Commands

Sample code in Cypher script

What is the Schema?

```
CALL db.schema.visualization
```

What are the relationship types?

```
CALL db.relationshipTypes()
```

Display all nodes with their relationships (I)

```
MATCH (n) RETURN n
```

Display all nodes with their relationships (II)

```
MATCH (a)-[r]-() RETURN a, r
```

Commonly Used Commands

From last time

What are the node types?

```
CALL db.schema.nodeTypeProperties
```

What are the relationship types?

```
CALL db.relationshipTypes()
```

Display all nodes

```
MATCH (n) RETURN n
```

Who reviewed what?

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

Who produced what?

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

Orchestral Connections

New Example! New Example! New Example! New Example!

New Example!

New Example! New Example! New Example! New Example

But this first!

Do not copy and paste this code all at once into Neo4j. All node creation code goes in own field in Neo4j, then the edge creation code follows in the next field.

Or just copy and paste from the build file ...

```
Build file:  sandbox/orchestralBuild.txt
```

Orchestral Connections

Note: all node and edge code is to be in a single copy-paste

Clear away previous graph: past into own field in Neo4j

```
MATCH (n) DETACH DELETE (n)
```

Create nodes!

```
CREATE(
 :Woodwinds {
   name:"windPlayer",
   instrument:"clarinet"} )

CREATE(
 :Percussions {
  name:"PercussionPlayer",
  instrument:"Drum"} )
```

```
MATCH (n) RETURN n
```

Orchestral Connections

Create more nodes!!

```
CREATE(
 :Strings {
  name:"StringPlayers",
  instrument_1:"guitar",
  instrument_2:"violin"} )

CREATE(
 :Audience {
  name:"Listener"} )

CREATE(
 :Conductor {
  name: "Conductor",
  instrument_1:"baton"} )
```

```
MATCH (n) RETURN n
```

Orchestral Connections

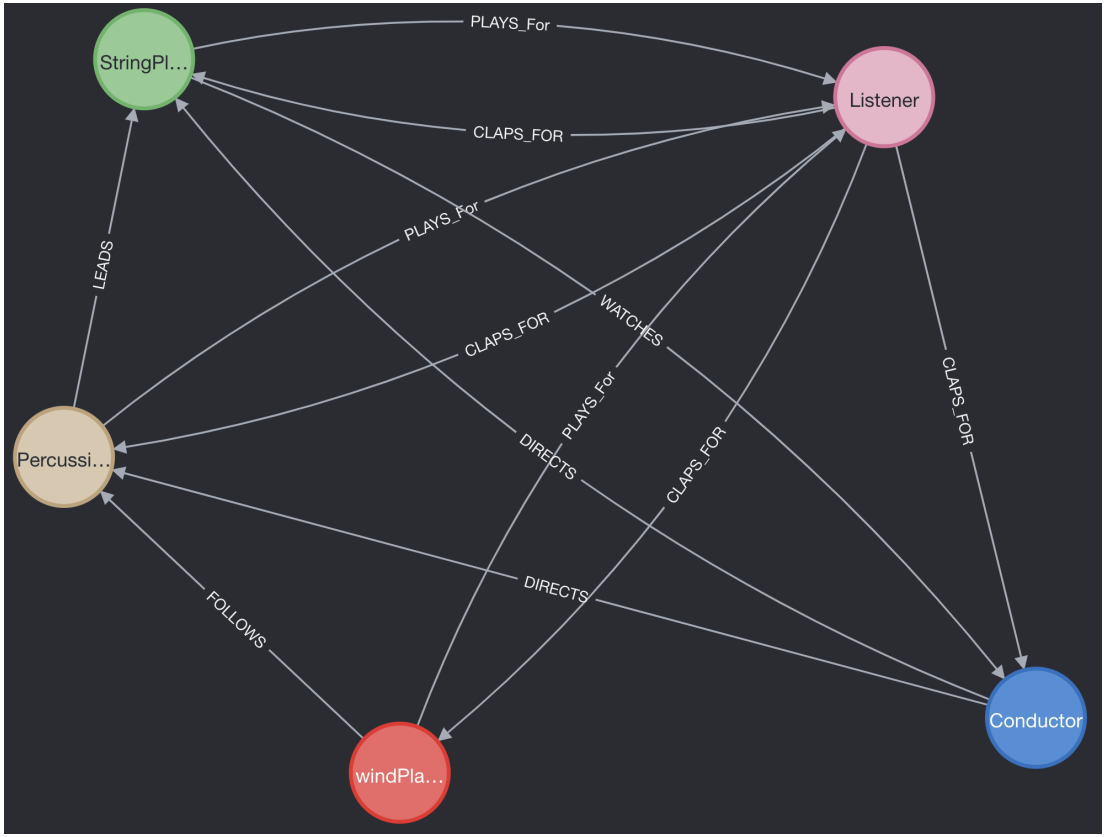
Define Node Variables and Edges

```
MATCH (w:Woodwinds), (p:Percussions),
  (s: Strings), (a:Audience), (c:Conductor)

MERGE (w) - [:FOLLOWS] -> (p) <-[:DIRECTS] - (c)
MERGE (p) - [:LEADS] -> (s) <-[:DIRECTS] - (c)
MERGE (s) - [:WATCHES] -> (c)

MERGE (w) - [:PLAYS_For] -> (a)
MERGE (p) - [:PLAYS_For] -> (a)
MERGE (s) - [:PLAYS_For] -> (a)

MERGE (a) - [:CLAPS_FOR] -> (w)
MERGE (a) - [:CLAPS_FOR] -> (p)
MERGE (a) - [:CLAPS_FOR] -> (s)
MERGE (a) - [:CLAPS_FOR] -> (c)
```

Show the graph

```
MATCH (n) RETURN (n)
```

Orchestral Connections



What is the Schema?

```
CALL db.schema.visualization
```

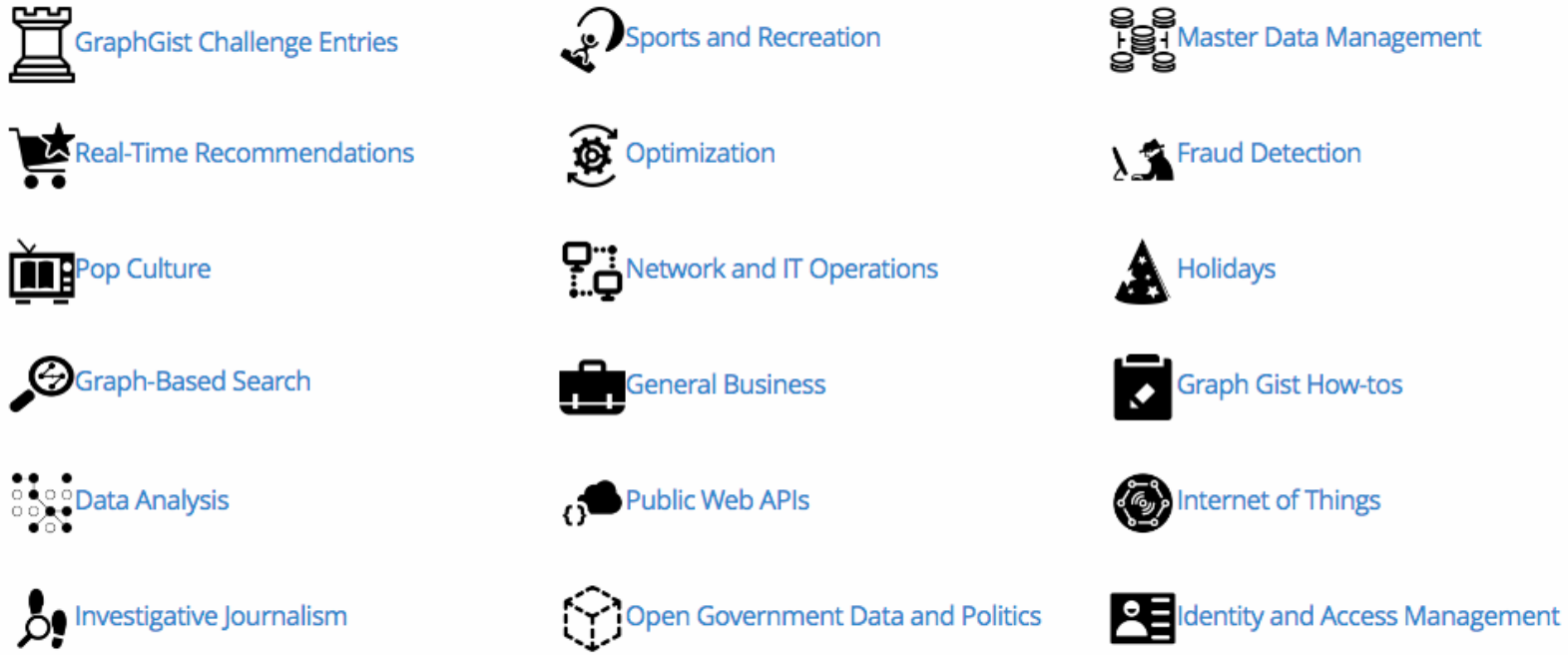
Check out this tutorial ...

First Steps with Cypher

`https://neo4j.com/graphgists/first-steps-with-cypher/`

Note: Be sure to use your local installation of Neo4J at `http://localhost:7474/browser/` to run your experiments by copying and pasting code from the tutorial.

Spend Some Time Playing With Other Graphs ...



- See What the community has done with Neo4j
- Graphgists Projects: `https://neo4j.com/graphgists/`

How To Shut Down a Session

Stop Neo4j container

```
docker stop testneo4j # Windows
sudo docker stop testneo4j # MacOS and Linux
```

Consider This...

- Can you work with data as nodes and edges in *your own* network?
- Can you discover new relationships between your nodes?