Introduction to Database Systems: CS312 Neo4J; Building Your Own Graph

Oliver Bonham-Carter

17 April 2019

Databases, Visually

Neo4J
Start Neo4J
Cypher Code
Graphgists Projects
Shutting Down
Consider this

- 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

- Copy your Neo4J setup tar file (neo4j-community-3.5.4-unix.tar.gz) to a desktop directory. Do not copy this file or open it in your submission directory!!!
- Click on this file to unpack its contents
- Locate the `bin` directory and then locate file: `neo4j`

Start the Neo4J server

```
./neo4j start
```

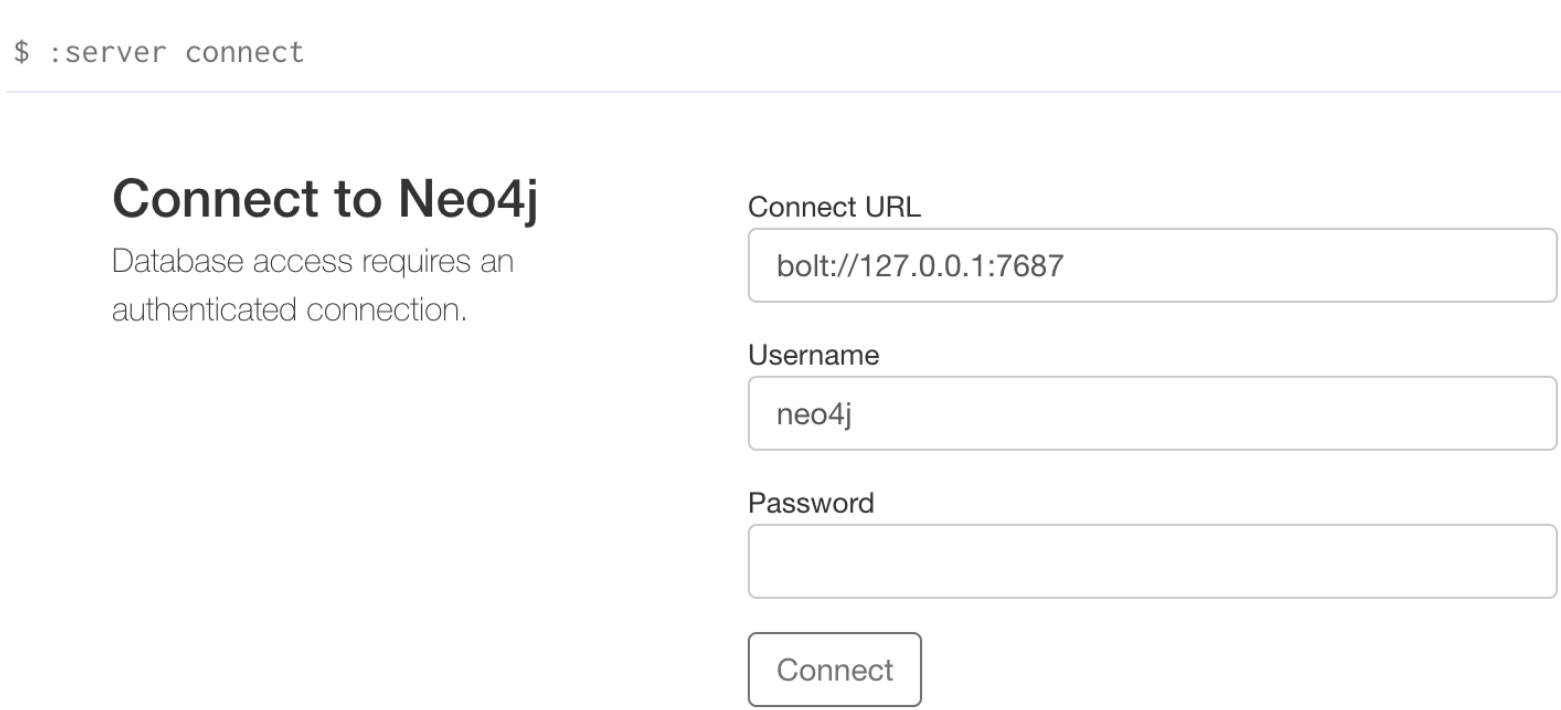
Check the status

```
./neo4j status
```

Note: Stop the Neo4J server

```
./neo4j stop
```

Login



- Open your browser and head to: `http://127.0.0.1:7474/browser/`
- Your first login
  - `User: neo4j`
  - `Password: neo4j`
- (You should have already changed your password by this time...)

Add Nodes

Destroy all nodes in the graph and erase the graph

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

Add the nodes

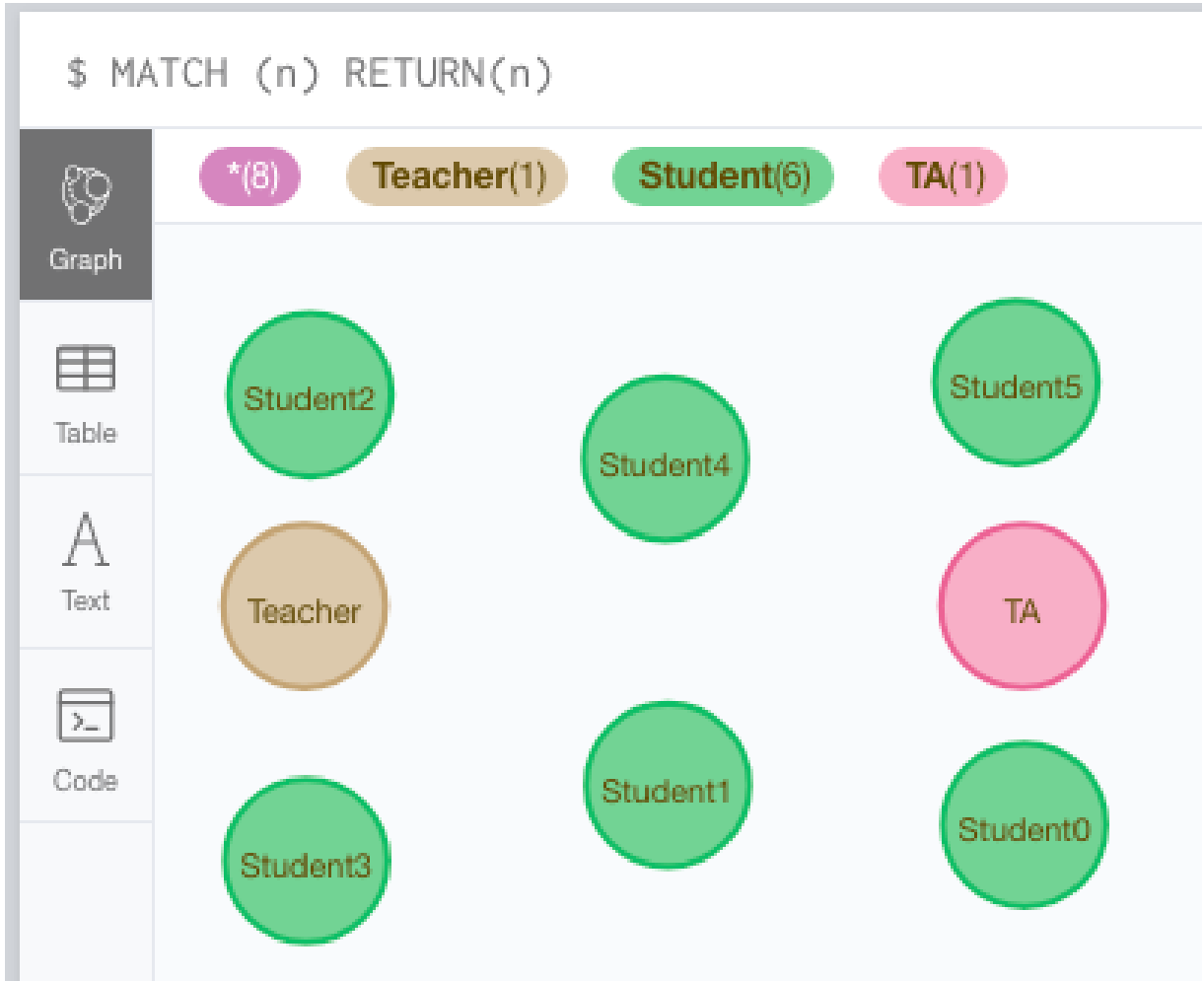
```
CREATE (
  :Teacher {
  name: "Teacher",
  Jackjet: "green",
  Jeans: "blue",
  MarkerCol: "red"}
)
FOREACH (r IN range(0,5)|
   CREATE (
     :Student { name:"Student" + r,
     extraUtility: "backpack" + r,
     lastTestScore:tan(rand())*100 })
     )
CREATE (:TA { name: "TA", Machine: "Laptop"})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

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

Show the Nodes

Show the unconnected graph

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



Add Edges

Add some connectivity to nodes

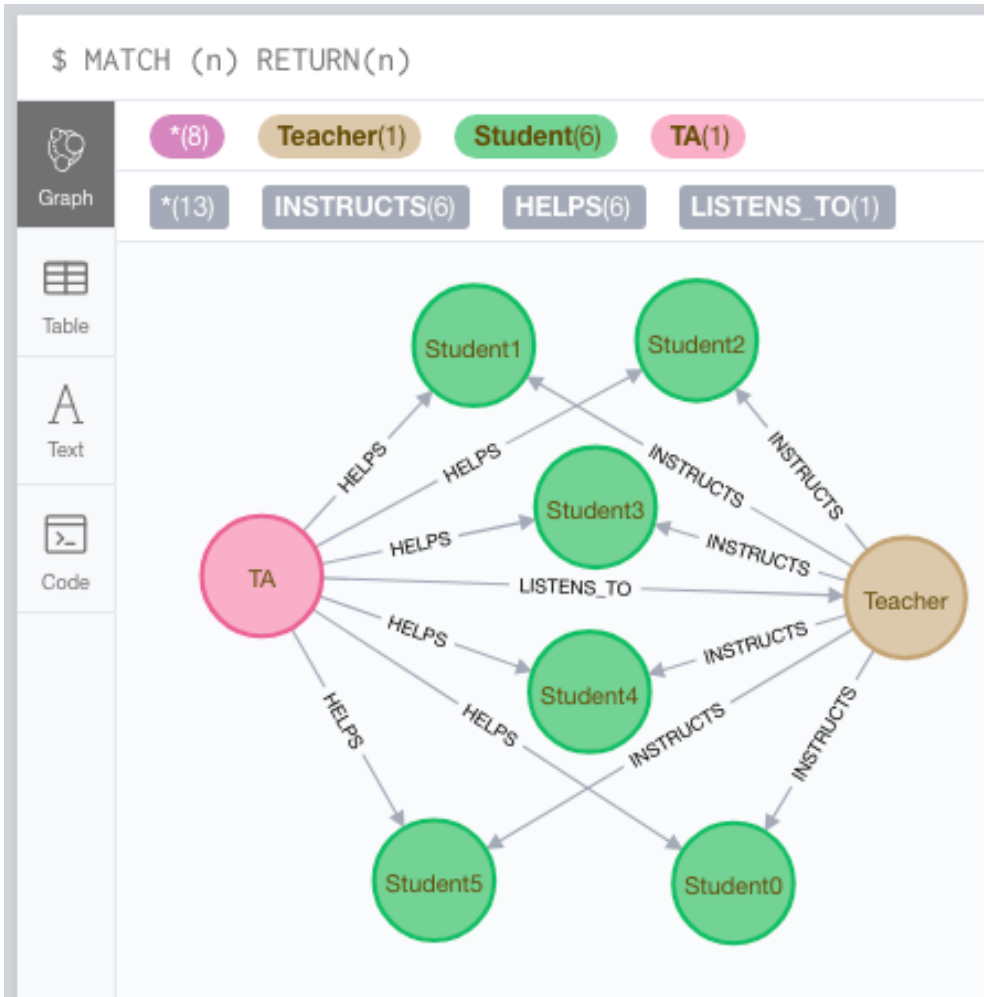
```
MATCH (t:Teacher), (s:Student), (a:TA)
CREATE UNIQUE (t) - [:INSTRUCTS] -> (s) <-[:HELPS] - (a)
CREATE UNIQUE (a) - [:LISTENS_TO] -> (t)
```

- 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, $->$

Show the Edges

Show the connected graph

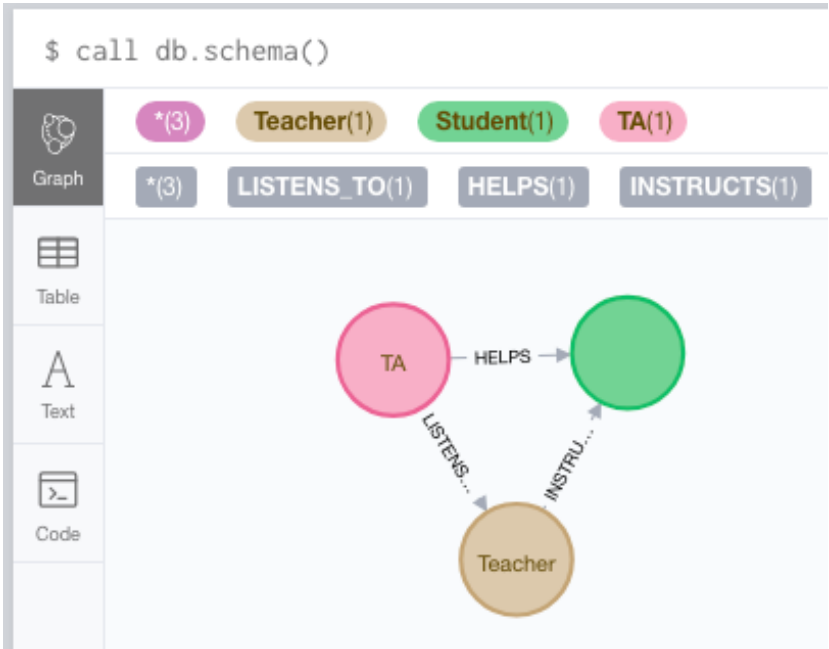
```
MATCH (n) RETURN (n)
```



Schema

Show the schema

```
call db.schema()
```

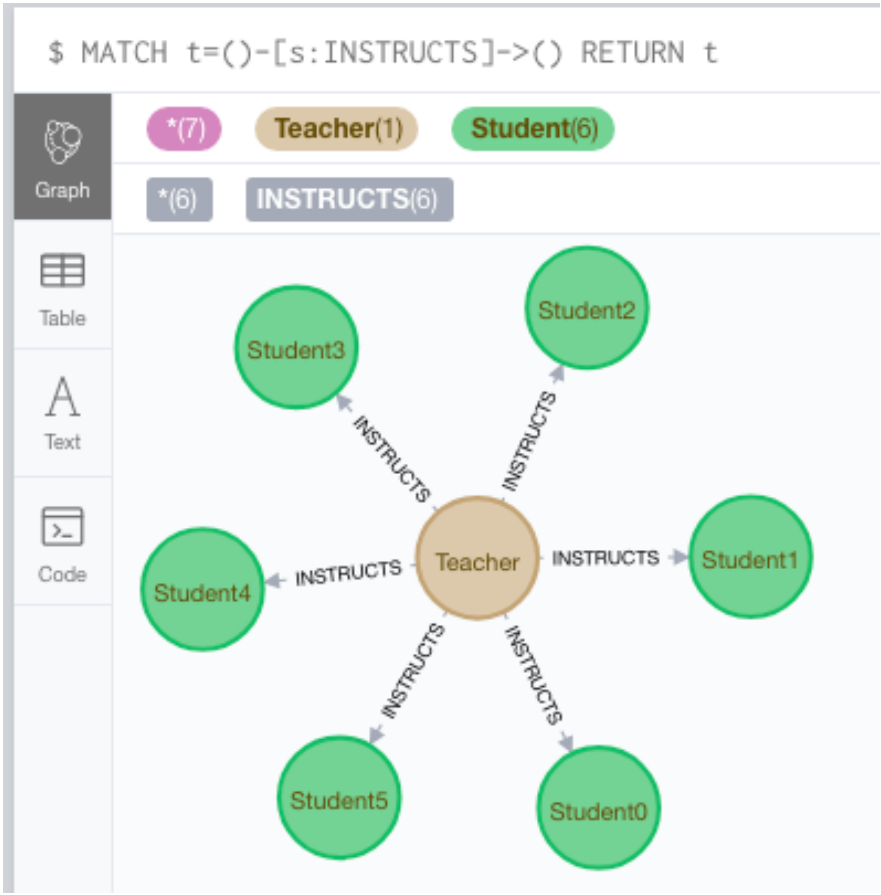


- The `Teacher` Instructs each `Student`
- The `TA` helps each `Student`
- The `TA` listens to the `Teacher`

Relationship Queries

Who Instructs whom?

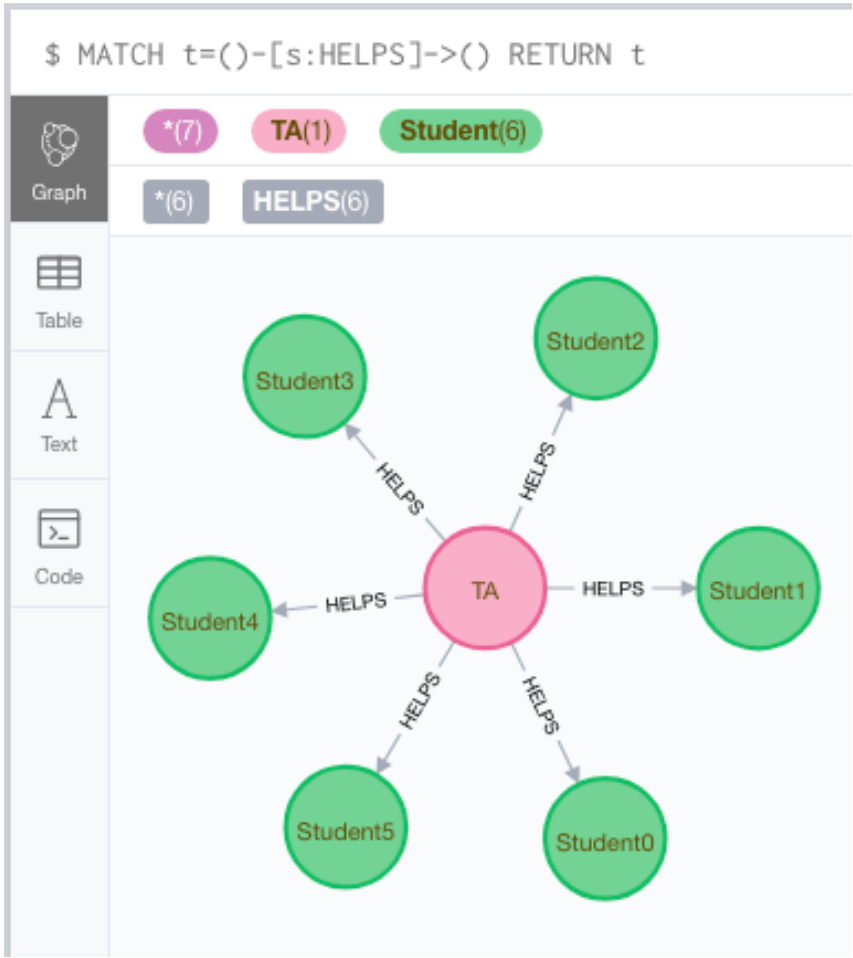
```
MATCH t=()-[s:INSTRUCTS]->() RETURN t
```



Relationship Queries

Who helps whom?

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

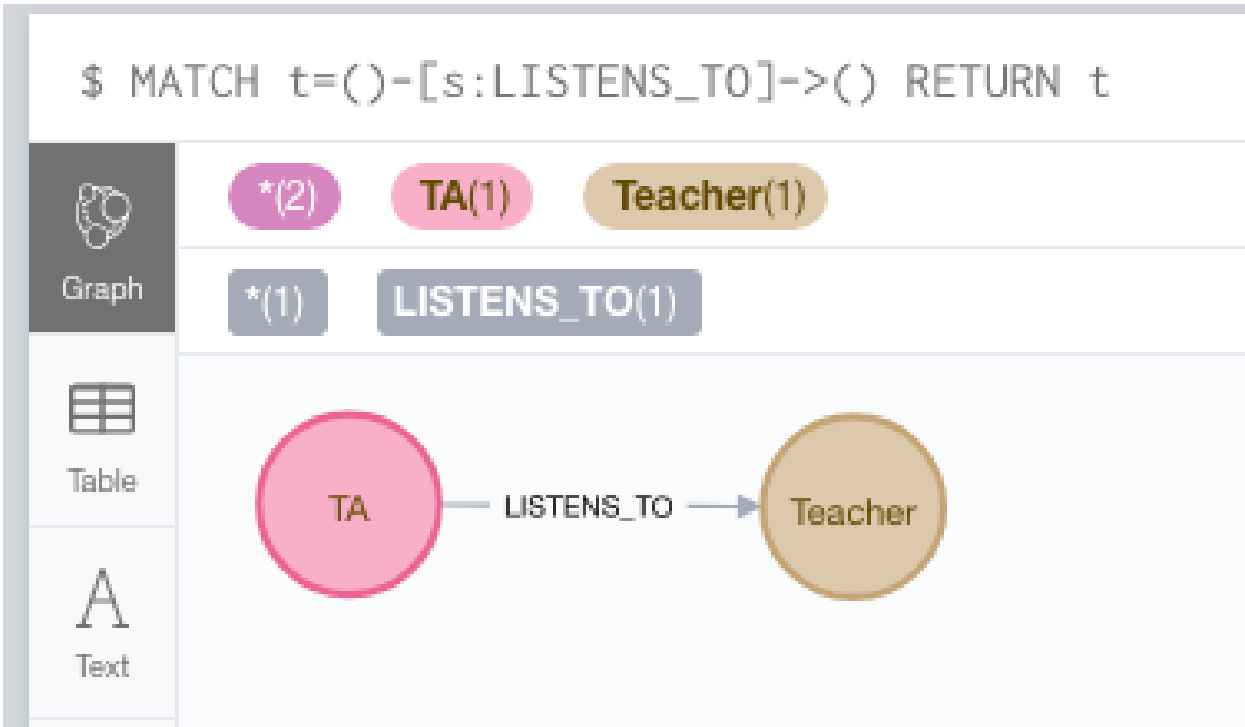


Relationship Queries

Neo4J
Start Neo4J
Cypher Code
Common Commands
Graphgists Projects
Shutting Down
Consider this

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()
```

What are the relationship types?

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

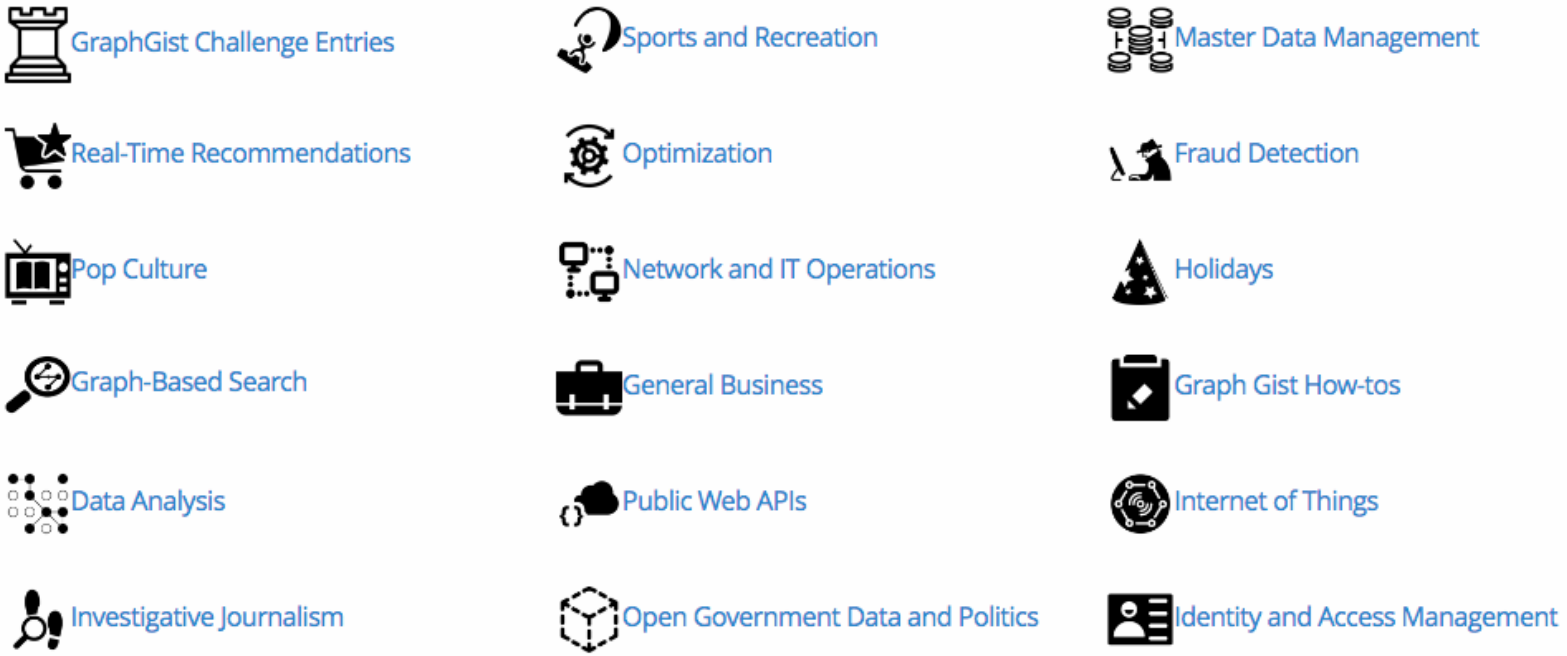
Display all nodes with their relationships (I)

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

Display all nodes with their relationships (II)

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

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

Neo4J
Start Neo4J
Cypher Code
Graphgists Projects
Shutting Down
Consider this

Closing down

- ./neo4j stop
- ./neo4j status

Consider this...

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