

# Class 7 Neo4j

## Elements of Databases

Apr 8, 2022

# Instapolls

- GCP credit balance
- Neo4j setup

# Announcements

## Exam 1 feedback:

- T/F and MC were fine
- Coding was challenging
- Not enough time

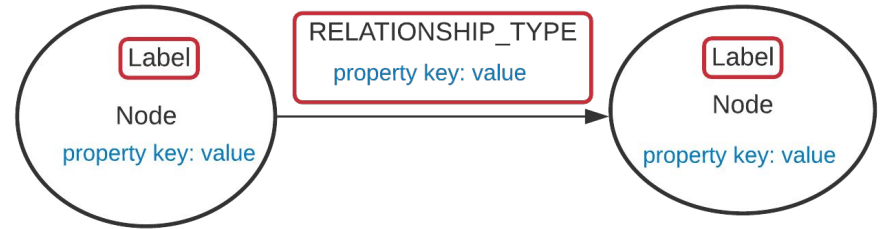
## Exam 2:

- Same format
- Fewer coding questions
- Review session next class
- Exam in two classes (April 22 at 4pm)

## Exam rules:

- Open-note and open-book
- Piazza will be disabled for public posts
- Piazza will be enabled for private posts to instructors
- May **not** consult or get help from anyone during exam

# Neo4j Overview

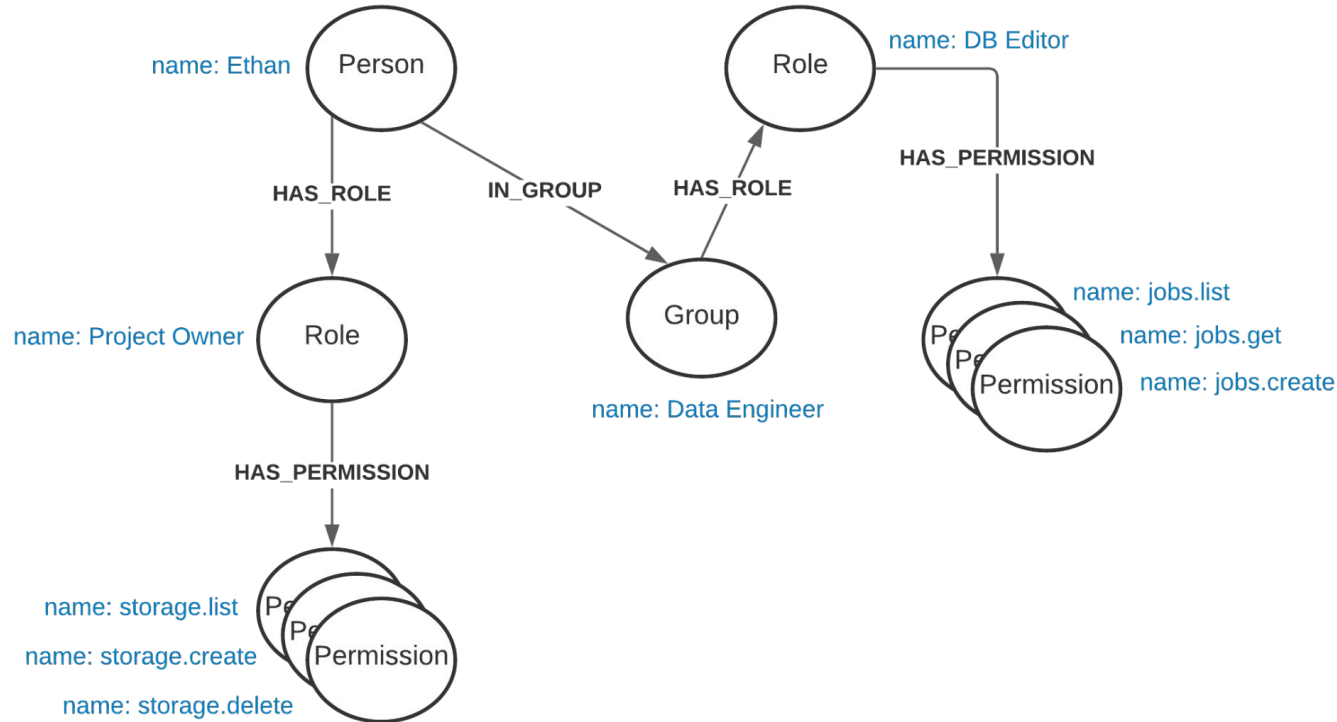


- + Labeled property graph database
- + Highly connected data
- + Declarative, SQL-inspired query language (Cypher)
- + Open-source, sponsored by Neo4j Inc.
- + Rich plugin and extension language (similar to Postgres)
- + ACID-compliant transactions
- + Distributed architecture for scaling reads
- + Visualization tools (Neo4j Browser, Bloom)
- + Optimized for graph traversals
- + Available as a cloud offering (Aura)
- Limited scalability for writes (no sharding)

# “Hello World” in Cypher

```
1 CREATE ();
2 CREATE (:Person);
3 CREATE (:Place);
4
5 MATCH(n) RETURN n;
6
7 CREATE (:Person {name: "Ethan"})-[:LIVES_IN]->(:Place {city: "Austin"});
8 CREATE (:Person {name: "Sasha"})-[:LIVES_IN]->(:Place {city: "New York"});
9
10 MATCH (p)-[r]->(c)
11 RETURN p, type(r), c;
12
13 MATCH ()-[r]->()
14 RETURN type(r), COUNT(r);
15
16 MATCH (p)-[:LIVES_IN]->(c)
17 WHERE p.name = "Ethan"
18 AND c.city = "Austin"
19 RETURN p, r, c;
```

# IAM model: a labeled property graph example



# Creating the IAM Nodes

```
1 CREATE (:Person {name: "Ethan", email: "ethan@utexas.edu"});
2 CREATE (:Group {name: "Data Engineer", owner: "Alex"});
3
4 CREATE (:Role {name: "Owner", resource: "Project"});
5 CREATE (:Role {name: "DB Editor", resource: "Cloud SQL"});
6
7 CREATE (:Permission {name: "jobs.list"});
8 CREATE (:Permission {name: "jobs.get"});
9 CREATE (:Permission {name: "jobs.create"});
10
11 CREATE (:Permission {name: "storage.list"});
12 CREATE (:Permission {name: "storage.create"});
13 CREATE (:Permission {name: "storage.delete"});
```

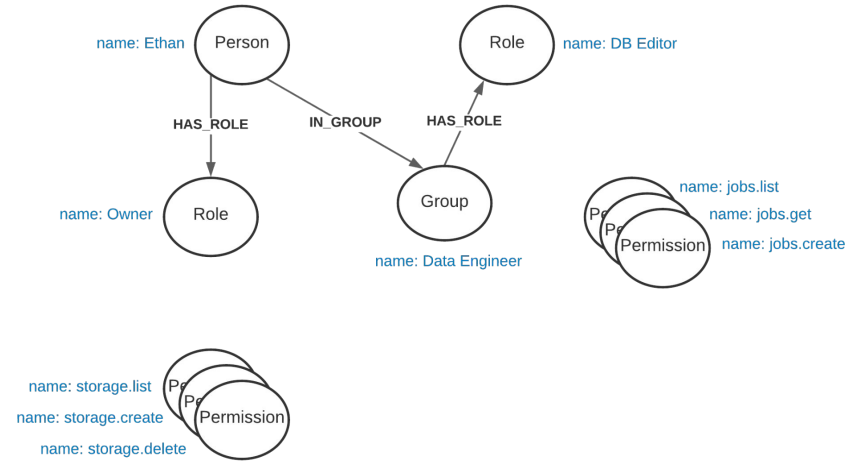
```
+-----+
| n      |
+-----+
| (:Person {name: "Ethan", email: "ethan@utexas.edu"}) |
| (:Group {owner: "Alex", name: "Data Engineer"})      |
| (:Role {name: "Owner", resource: "Project"})         |
| (:Role {name: "DB Editor", resource: "Cloud SQL"})   |
| (:Permission {name: "jobs.list"})                   |
| (:Permission {name: "jobs.get"})                    |
| (:Permission {name: "jobs.create"})                  |
| (:Permission {name: "storage.list"})                 |
| (:Permission {name: "storage.create"})               |
| (:Permission {name: "storage.delete"})               |
+-----+
```

# Creating the Relationships

```

1 MATCH (p:Person {name: "Ethan"})
2 MATCH (r:Role {name: "Owner"})
3 CREATE (p)-[:HAS_ROLE]->(r);
4
5 MATCH (p:Person {name: "Ethan"})
6 MATCH (g:Group {name: "Data Engineer"})
7 CREATE (p)-[:IN_GROUP]->(g);
8
9 MATCH (g:Group {name: "Data Engineer"})
10 MATCH (r:Role {name: "DB Editor"})
11 CREATE (g)-[:HAS_ROLE]->(r);
12
13 MATCH (p)-[h]->(r) RETURN p, h, r;

```

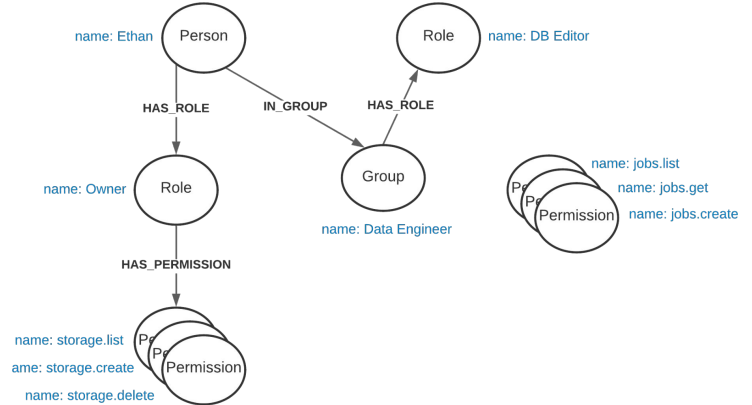


p	h	r
(:Person {name: "Ethan", email: "ethan@utexas.edu"})	[:IN_GROUP]	(:Group {owner: "Alex", name: "Data Engineer"})
(:Person {name: "Ethan", email: "ethan@utexas.edu"})	[:HAS_ROLE]	(:Role {name: "Owner", resource: "Project"})
(:Group {owner: "Alex", name: "Data Engineer"})	[:HAS_ROLE]	(:Role {name: "DB Editor", resource: "Cloud SQL"})



# Creating the Relationships (cont.)

```
16 MATCH (r:Role {resource: "Project"})
17 MATCH (p:Permission {name: "storage.list"})
18 CREATE (r)-[:HAS_PERMISSION]->(p);
19
20 MATCH (r:Role {resource: "Project"})
21 MATCH (p:Permission {name: "storage.create"})
22 CREATE (r)-[:HAS_PERMISSION]->(p);
23
24 MATCH (r:Role {name: "Owner"})
25 MATCH (p:Permission {name: "storage.delete"})
26 CREATE (r)-[:HAS_PERMISSION]->(p);
27
28 MATCH (r:Role)-[h]->(p:Permission)
29 WHERE r.resource = "Project" OR r.name = "Owner"
30 RETURN r, h, p;
```



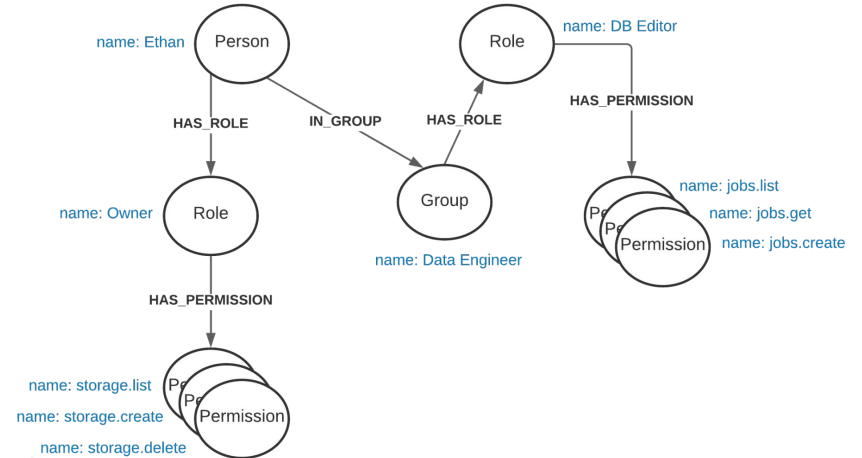
r	h	p
(:Role {name: "Owner", resource: "Project"})	[:HAS_PERMISSION]	(:Permission {name: "storage.delete"})
(:Role {name: "Owner", resource: "Project"})	[:HAS_PERMISSION]	(:Permission {name: "storage.create"})
(:Role {name: "Owner", resource: "Project"})	[:HAS_PERMISSION]	(:Permission {name: "storage.list"})

# Creating the Relationships (cont.)

```

33 MATCH (r:Role {name: "DB Editor"})
34 MATCH (p:Permission {name: "jobs.list"})
35 CREATE (r)-[:HAS_PERMISSION]->(p);
36
37 MATCH (r:Role {name: "DB Editor"})
38 MATCH (p:Permission {name: "jobs.get"})
39 CREATE (r)-[:HAS_PERMISSION]->(p);
40
41 MATCH (r:Role {name: "DB Editor"})
42 MATCH (p:Permission {name: "jobs.create"})
43 CREATE (r)-[:HAS_PERMISSION]->(p);
44
45 MATCH (r:Role)-[h:HAS_PERMISSION]->(p:Permission)
46 WHERE r.name = "DB Editor"
47 RETURN r, h, p;

```



r	h	p
(:Role {name: "DB Editor", resource: "Cloud SQL"})	[:HAS_PERMISSION]	(:Permission {name: "jobs.create"})
(:Role {name: "DB Editor", resource: "Cloud SQL"})	[:HAS_PERMISSION]	(:Permission {name: "jobs.get"})
(:Role {name: "DB Editor", resource: "Cloud SQL"})	[:HAS_PERMISSION]	(:Permission {name: "jobs.list"})

# Visualizing the Graph

localhost:7474/browser/

**Database Information**

Use database  
neo4j

Node Labels  
\*(10) Group Permission  
Person Role

Relationship Types  
\*(9) HAS\_PERMISSION  
HAS\_ROLE IN\_GROUP

Property Keys  
city email name owner  
resource

Connected as  
Username: neo4j  
Roles: -  
Disconnect: :server disconnect

```
neo4j$
```

```
neo4j$ MATCH (n) RETURN n LIMIT 25
```

\*(10) Person(1) Group(1) Role(2) Permission(6)


\*(9) IN\_GROUP(1) HAS\_ROLE(2) HAS\_PERMISSION(6)

```
graph LR; Owner((Owner)) -- HAS_PERMISSION --> storage1((storage.I...)); Owner -- HAS_PERMISSION --> storage2((storage....)); Owner -- HAS_PERMISSION --> storage3((storage....)); Ethan((Ethan)) -- HAS_ROLE --> Owner; Ethan -- IN_GROUP --> DataEngineer((Data Engineer)); DataEngineer -- HAS_ROLE --> DBEditor((DB Editor)); DBEditor -- HAS_PERMISSION --> jobsCreate((jobs.cre...)); DBEditor -- HAS_PERMISSION --> jobsGet((jobs.get)); DBEditor -- HAS_PERMISSION --> jobsList((jobs.list));
```


Permission <id>: 6 name: jobs.create

# Counting Nodes and Relationships

```
1 MATCH (n)
2 RETURN count(n);
3
4 MATCH (n)
5 RETURN distinct labels(n), count(n);
6
7 MATCH ()-[r]->()
8 RETURN count(r);
9
10 MATCH ()-[r]->()
11 RETURN type(r), count(r);
12
13 MATCH (n:Person)
14 RETURN count(n);
15
16 MATCH ()-[r:HAS_ROLE]->()
17 RETURN count(r);
```



labels(n)	count(n)
["Person"]	1
["Group"]	1
["Role"]	2
["Permission"]	6



type(r)	count(r)
"IN_GROUP"	1
"HAS_ROLE"	2
"HAS_PERMISSION"	6

# Querying the Graph

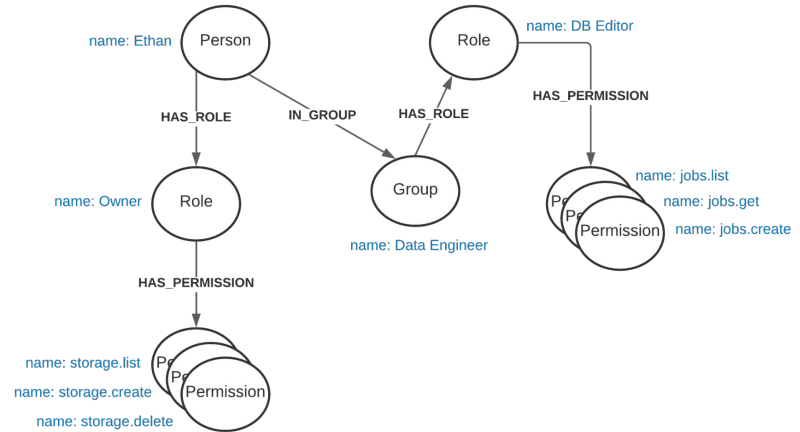
```
1 MATCH (p:Person)-[r*]->(m:Permission)
2 WHERE p.name = "Ethan"
3 RETURN r, m.name
4 ORDER BY m;
```

r	m.name
[[:IN_GROUP], [:HAS_ROLE], [:HAS_PERMISSION]]	"jobs.list"
[[:IN_GROUP], [:HAS_ROLE], [:HAS_PERMISSION]]	"jobs.get"
[[:IN_GROUP], [:HAS_ROLE], [:HAS_PERMISSION]]	"jobs.create"
[[:HAS_ROLE], [:HAS_PERMISSION]]	"storage.list"
[[:HAS_ROLE], [:HAS_PERMISSION]]	"storage.create"
[[:HAS_ROLE], [:HAS_PERMISSION]]	"storage.delete"

```
6 MATCH (p:Person)-[r*]->(m:Permission)
7 WHERE p.name = "Ethan"
8 WITH distinct m.name as distinct_perms
9 RETURN distinct_perms
10 ORDER BY distinct_perms;
```



distinct_perms
"jobs.create"
"jobs.get"
"jobs.list"
"storage.create"
"storage.delete"
"storage.list"



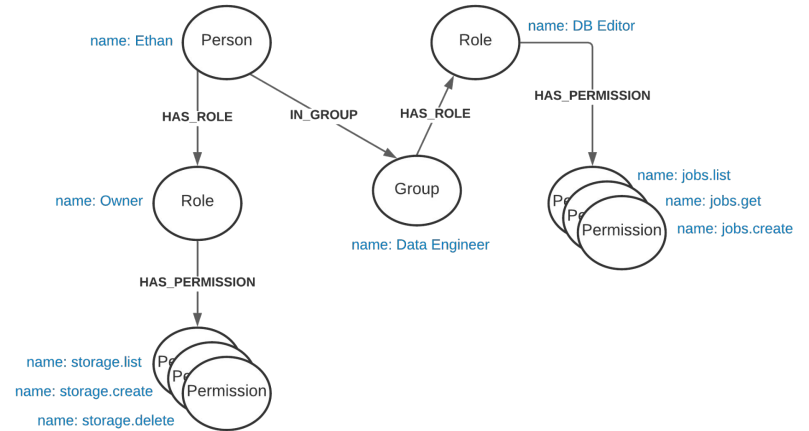
# Querying the Graph

```
12 MATCH (p:Person)-[r*1]->(m:Permission)
13 WHERE p.name = "Ethan"
14 RETURN r, m.name
15 ORDER BY m;
```

```
+-----+
| r | m.name |
+-----+
+-----+
```

```
17 MATCH (p:Person)-[r*1..2]->(m:Permission)
18 WHERE p.name = "Ethan"
19 RETURN r, m.name
20 ORDER BY m;
```

```
+-----+
| r | m.name |
+-----+
| [[:HAS_ROLE], [:HAS_PERMISSION]] | "storage.list" |
| [[:HAS_ROLE], [:HAS_PERMISSION]] | "storage.create" |
| [[:HAS_ROLE], [:HAS_PERMISSION]] | "storage.delete" |
+-----+
```



# Updating Nodes

## Adding node properties:

```
1 MATCH (n:Person {name: "Ethan"})
2 SET n.current_employee = True,
3   n.start_date = "2021-06-01"
4 RETURN n.name, n.current_employee, n.start_date;
5
6 MATCH (n:Person {name: "Ethan"})
7 SET n.current_employee = False,
8   n.start_date = "2021-06-01",
9   n.end_date = "2021-08-01"
10 RETURN n.name, n.current_employee, n.start_date, n.end_date;
```



n.name	n.current_employee	n.start_date
"Ethan"	TRUE	"2021-06-01"



n.name	n.current_employee	n.start_date	n.end_date
"Ethan"	FALSE	"2021-06-01"	"2021-08-01"

## Adding node labels:

```
12 MATCH (n {name: "Ethan"})
13 SET n:Principal
14 RETURN n.name, labels(n) AS labels;
```



n.name	labels
"Ethan"	["Person", "Principal"]



# Updating Relationships

## Adding and updating relationship properties:

```
16 MATCH (n:Role {name: "DB Editor"})
17 MATCH (p:Permission {name: "jobs.create"})
18 MERGE (n)-[r:HAS_PERMISSION]->(p)
19 ON MATCH SET r.start_time = "08:00", r.end_time = "17:00"
20 RETURN n.name, type(r), r.start_time, r.end_time;
```

n.name	type(r)	r.start_time	r.end_time
"DB Editor"	"HAS_PERMISSION"	"08:00"	"17:00"

## "Renaming" a relationship type:

```
22 MATCH (n:Role)-[rel:HAS_PERMISSION]->(p:Permission)
23 MERGE (n)-[:HAS_IAM_PERMISSION]->(p)
24 DELETE rel;
25
26 MATCH (r:Role)-[h:HAS_IAM_PERMISSION]->(p:Permission)
27 RETURN r, h, p;
```

r	h	p
{:Role {name: "Owner", resource: "Project"}}	[:HAS_IAM_PERMISSION]	{:Permission {name: "storage.list"}}
{:Role {name: "Owner", resource: "Project"}}	[:HAS_IAM_PERMISSION]	{:Permission {name: "storage.create"}}
{:Role {name: "Owner", resource: "Project"}}	[:HAS_IAM_PERMISSION]	{:Permission {name: "storage.delete"}}
{:Role {name: "DB Editor", resource: "Cloud SQL"}}	[:HAS_IAM_PERMISSION]	{:Permission {name: "jobs.list"}}
{:Role {name: "DB Editor", resource: "Cloud SQL"}}	[:HAS_IAM_PERMISSION]	{:Permission {name: "jobs.get"}}
{:Role {name: "DB Editor", resource: "Cloud SQL"}}	[:HAS_IAM_PERMISSION]	{:Permission {name: "jobs.create"}}



# Deleting Relationships and Nodes

Drop the relationships connected to nodes labeled Person:

```
1 MATCH (p:Person)-[r]->()  
2 DELETE r;
```

Drop nodes labeled Person:

```
4 MATCH (p:Person)  
5 DELETE p;
```

Drop all the nodes and relationships in the current database:

```
7 MATCH (n)  
8 DETACH DELETE n;
```

```
neo4j@neo4j> MATCH (n)  
          DETACH DELETE n;  
0 rows available after 7 ms, consumed after another 0 ms  
Deleted 10 nodes, Deleted 9 relationships  
neo4j@neo4j>
```

# Neo4j Code Lab

- Clone [snippets](#) repo
- Open [neo4j notebook](#)
- Create movie graph database
- Write cypher queries to explore the graph

# Practice Problem

Translate the following question into a Cypher query:

*Which persons acted in their own movie?*

*Return the person's name, the movie title, and the role they played in the movie which they directed.*

*Order the results by person's name.*

# Project 7

<http://www.cs.utexas.edu/~scohen/projects/Project7.pdf>

