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

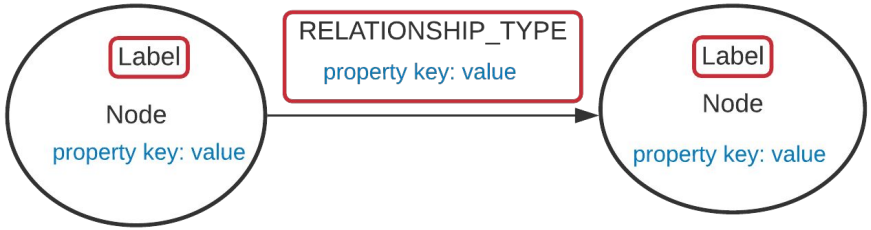
Exam rules:

- 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

Exam 2:

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

Neo4j Overview

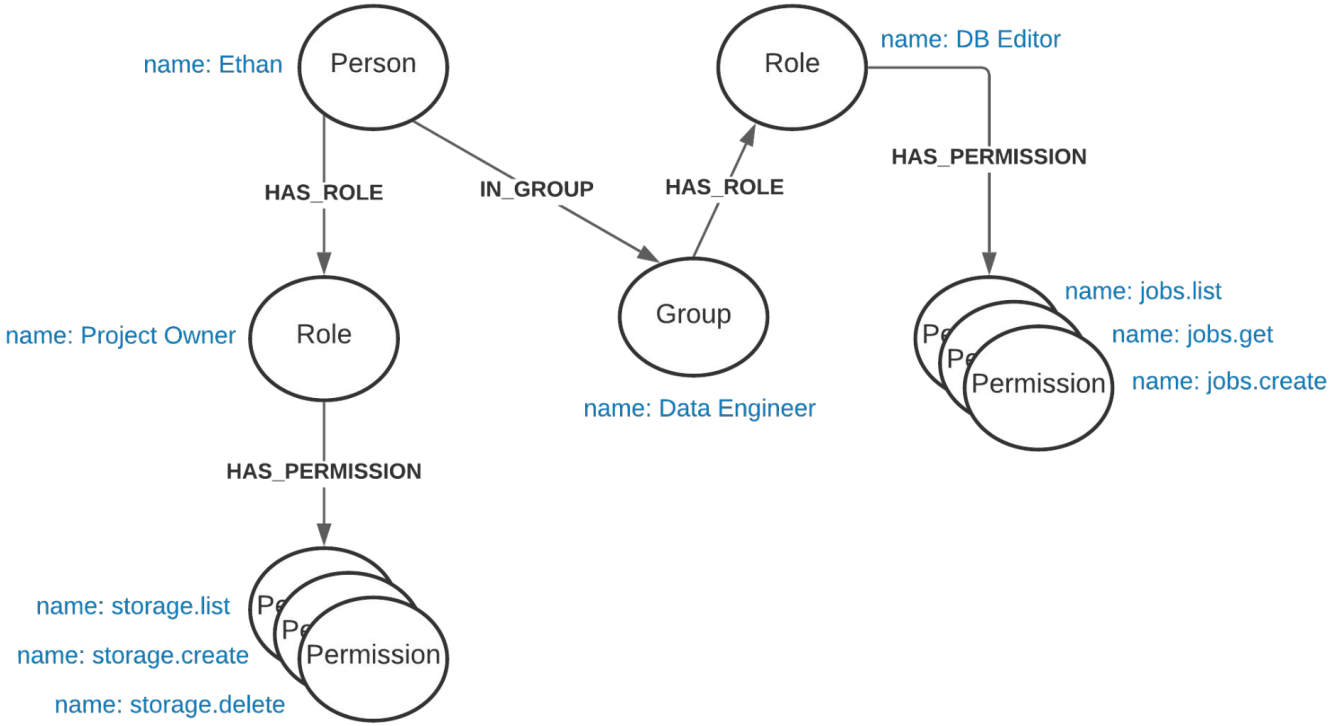


+ Labeled property graph database
+ Highly connected data
+ Declarative, SQL-inspired query language (Cypher)
+ Open-source, sponsored by Neo4j Inc.
+ Rich plugin and extension language (similar to Postgres)
+ ACID-compliant transactions
+ Distributed architecture for scaling reads
+ Visualization tools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

“Hello World” in Cypher

```
CREATE ();
CREATE (:Person);
CREATE (:Place);

MATCH(n) RETURN n;

CREATE (:Person {name: "Ethan"})-[:LIVES_IN]->(:Place {city: "Austin"});
CREATE (:Person {name: "Sasha"})-[:LIVES_IN]->(:Place {city: "New York"});

MATCH (p)-[r]->(c)
RETURN p, type(r), c;

MATCH ()-[r]->()
RETURN type(r), COUNT(r);

MATCH (p)-[r:LIVES_IN]->(c)
WHERE p.name = "Ethan"
AND c.city = "Austin"
RETURN p, r, c;
```

IAM model: a labeled property graph example



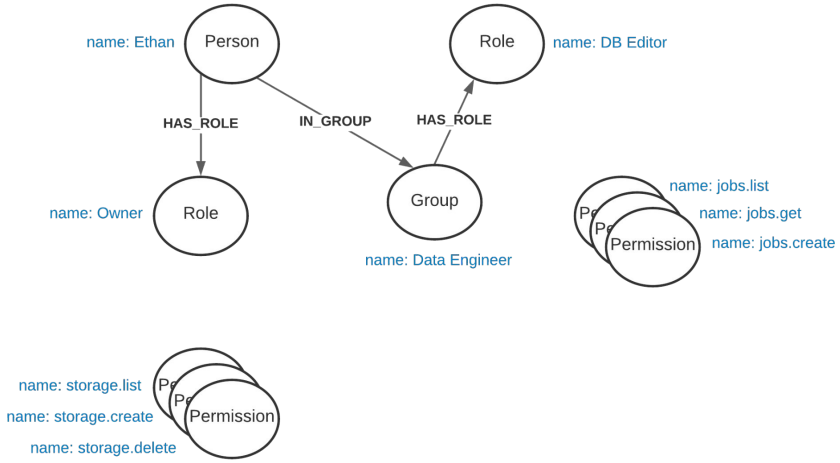
Creating the IAM Nodes

```
CREATE (:Person {name: "Ethan", email: "ethan@utexas.edu"});
CREATE (:Group {name: "Data Engineer", owner: "Alex"});

CREATE (:Role {name: "Owner", resource: "Project"});
CREATE (:Role {name: "DB Editor", resource: "Cloud SQL"});

CREATE (:Permission {name: "jobs.list"});
CREATE (:Permission {name: "jobs.get"});
CREATE (:Permission {name: "jobs.create"});

CREATE (:Permission {name: "storage.list"});
CREATE (:Permission {name: "storage.create"});
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

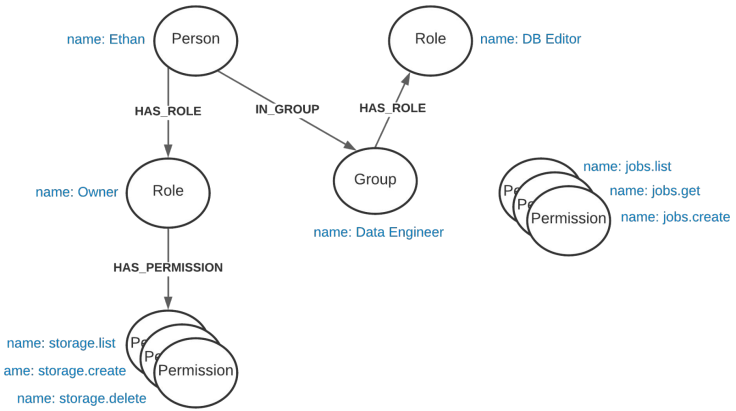
```
MATCH (p:Person {name: "Ethan"})
MATCH (r:Role {name: "Owner"})
CREATE (p)-[:HAS_ROLE]->(r);

MATCH (p:Person {name: "Ethan"})
MATCH (g:Group {name: "Data Engineer"})
CREATE (p)-[:IN_GROUP]->(g);

MATCH (g:Group {name: "Data Engineer"})
MATCH (r:Role {name: "DB Editor"})
CREATE (g)-[:HAS_ROLE]->(r);

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.)

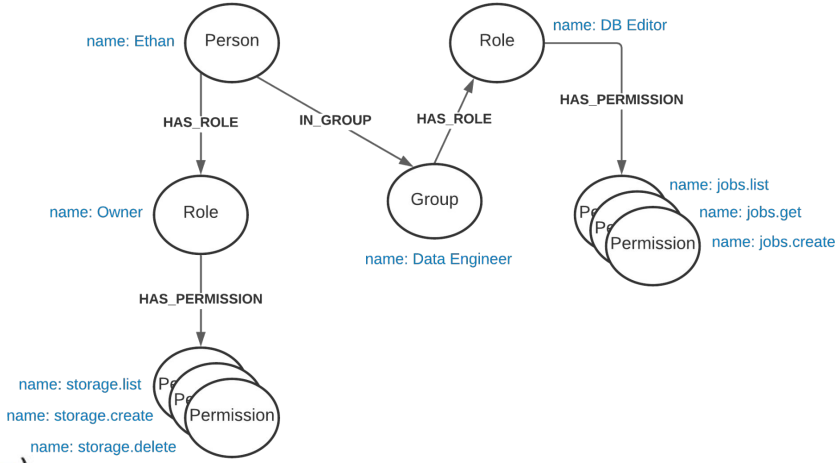
```
MATCH (r:Role {resource: "Project"})
MATCH (p:Permission {name: "storage.list"})
CREATE (r)-[:HAS_PERMISSION]->(p);

MATCH (r:Role {resource: "Project"})
MATCH (p:Permission {name: "storage.create"})
CREATE (r)-[:HAS_PERMISSION]->(p);

MATCH (r:Role {name: "Owner"})
MATCH (p:Permission {name: "storage.delete"})
CREATE (r)-[:HAS_PERMISSION]->(p);

MATCH (r:Role)-[h]->(p:Permission)
WHERE r.resource = "Project" OR r.name = "Owner"
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"})

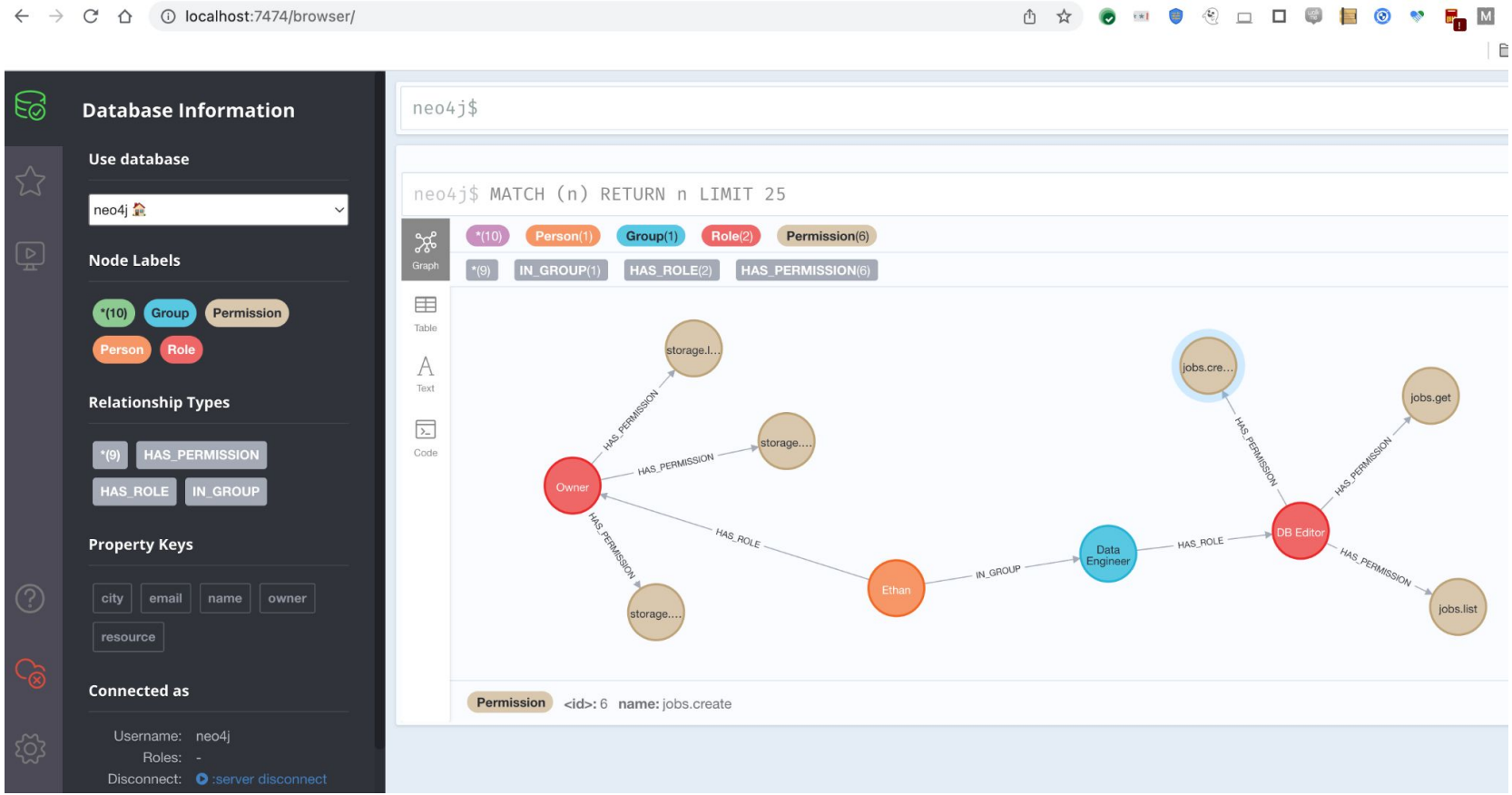
Creating the Relationships (cont.)

```
MATCH (r:Role {name: "DB Editor"})
MATCH (p:Permission {name: "jobs.list"})
CREATE (r)-[:HAS_PERMISSION]->(p);

MATCH (r:Role {name: "DB Editor"})
MATCH (p:Permission {name: "jobs.get"})
CREATE (r)-[:HAS_PERMISSION]->(p);

MATCH (r:Role {name: "DB Editor"})
MATCH (p:Permission {name: "jobs.create"})
CREATE (r)-[:HAS_PERMISSION]->(p);

MATCH (r:Role)-[h:HAS_PERMISSION]->(p:Permission)
WHERE r.name = "DB Editor"
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



Counting Nodes and Relationships

```
MATCH (n)
RETURN count(n);

MATCH (n)
RETURN distinct labels(n), count(n);

MATCH ()-[r]->()
RETURN count(r);

MATCH ()-[r]->()
RETURN type(r), count(r);

MATCH (n:Person)
RETURN count(n);

MATCH ()-[r:HAS_ROLE]->()
RETURN count(r);
```

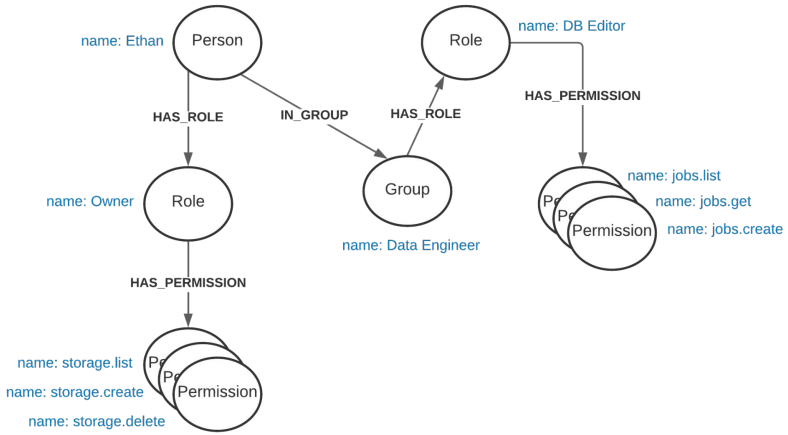
```
+---------------------------+
| labels(n)       | count(n) |
+---------------------------+
| ["Person"]      | 1        |
| ["Group"]       | 1        |
| ["Role"]        | 2        |
| ["Permission"]  | 6        |
+---------------------------+
```

```
+----------------------------+
| type(r)          | count(r) |
+----------------------------+
| "IN_GROUP"       | 1        |
| "HAS_ROLE"       | 2        |
| "HAS_PERMISSION" | 6        |
+----------------------------+
```

Querying the Graph

```
MATCH (p:Person)-[r*]->(m:Permission)
WHERE p.name = "Ethan"
RETURN r, m.name
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"




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

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

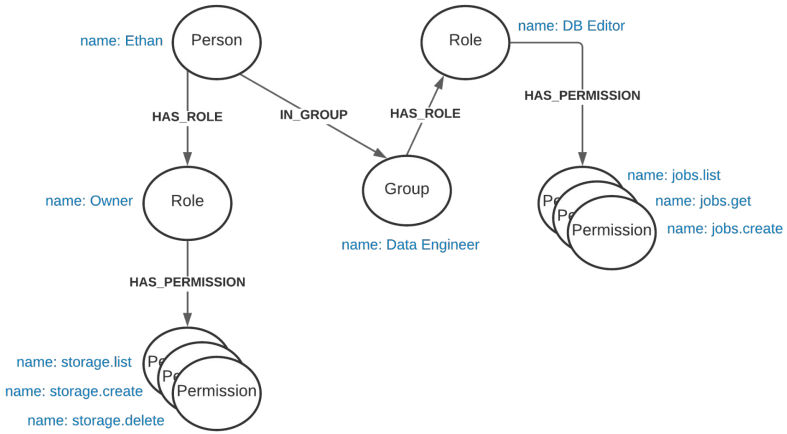
Querying the Graph

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

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

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

```
MATCH (n:Person {name: "Ethan"})
SET n.current_employee = True,
    n.start_date = "2021-06-01"
RETURN n.name, n.current_employee, n.start_date;

MATCH (n:Person {name: "Ethan"})
SET n.current_employee = False,
    n.start_date = "2021-06-01",
    n.end_date = "2021-08-01"
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:

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

```
+-----------------------------------+
| n.name   | labels                 |
+-----------------------------------+
| "Ethan" | ["Person", "Principal"] |
+-----------------------------------+
```

Updating Relationships

Adding and updating relationship properties:

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

```
MATCH (n:Role)-[rel:HAS_PERMISSION]->(p:Permission)
MERGE (n)-[:HAS_IAM_PERMISSION]->(p)
DELETE rel;

MATCH (r:Role)-[h:HAS_IAM_PERMISSION]->(p:Permission)
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:

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

Drop nodes labeled Person:

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

Drop all the nodes and relationships in the current database:

```
MATCH (n)
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