# CS 327E Class 7

March 26, 2021

# Announcements

- Test 2 next class at 4pm CT
- Review session on Wed from 3pm - 4pm CT

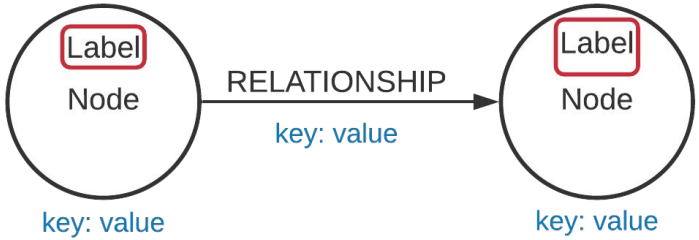
**Exam rules:**

- 60 minutes
- Open-note and open-book
- Piazza will be disabled during exam
- May **not** consult with any human in any form

# Instapolls

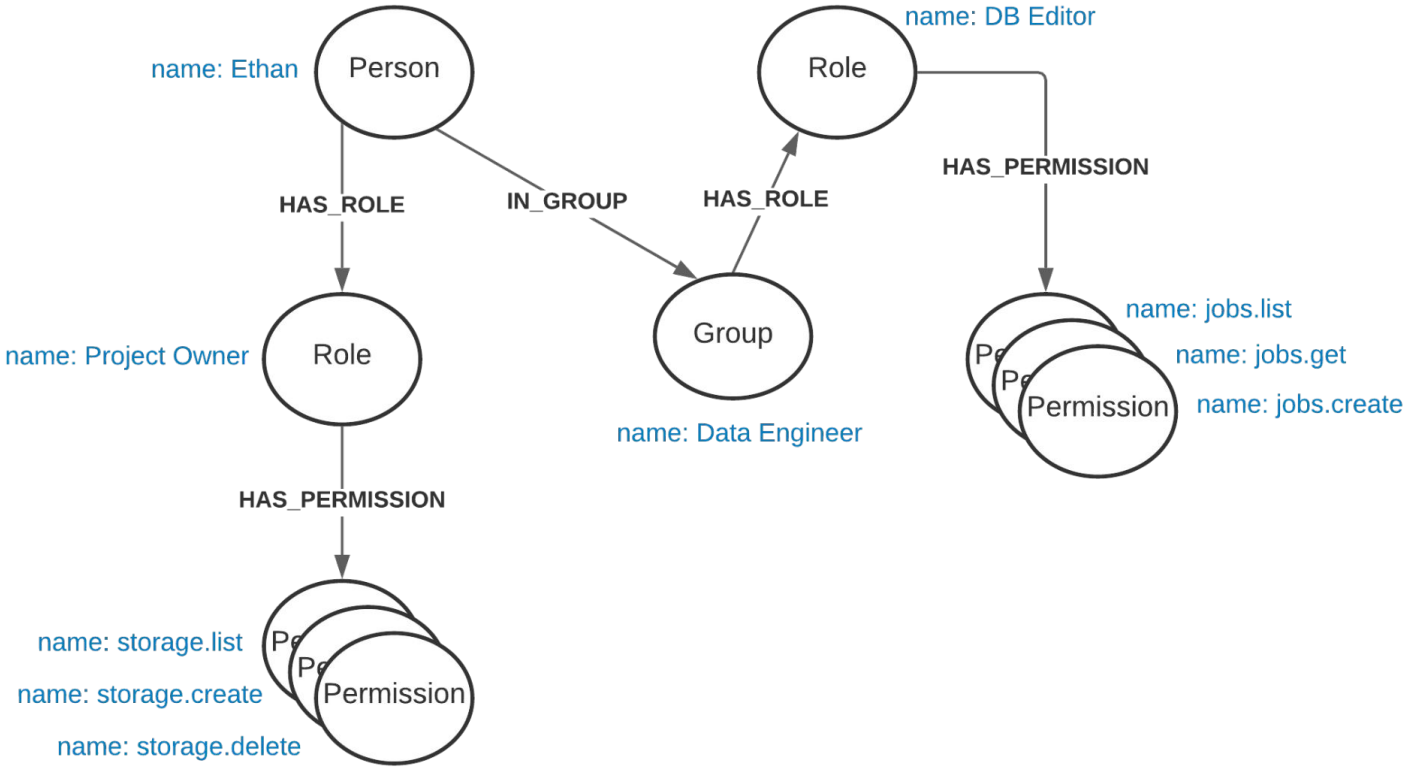
- Check your Neo4j set up
- Check your GCP credits

# Why Neo4j?




+ Labeled property graph data model
+ Flexible schema
+ 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
- No sharding
- Doesn’t come as a managed service in the cloud

# Neo4j’s Data Model



# “Hello World” in Cypher

```
CREATE ();
CREATE (:Person);
CREATE (:Place);
CREATE (:Person {name: "Brad"})-[:LIVES_IN]->(:Place {city: "Austin"});

MATCH(n) RETURN n;

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

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

# Create the Nodes

```
CREATE (:Person {name: "Ethan", email: "ethan@utexas.edu"});
CREATE (:Group {name: "Data Engineer", owner: "Alex"});
CREATE (:Role {name: "Project Owner", type: "GCP"});
CREATE (:Role {name: "DB Editor", type: "MySQL"});

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"});

MATCH (n) RETURN n;
```

# Connecting the Nodes

```
MATCH (p:Person {name: "Ethan"})
MATCH (r:Role {name: "Project 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;

MATCH (p:Person)-[h]->(r:Role)
WHERE r.type = "MySQL"
RETURN p, h, r;
```

# Connecting the Nodes

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

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

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

MATCH (r:Role)-[h]->(p)
WHERE r.name = "Project Owner"
RETURN r, h, p;
```

# Connecting the Nodes

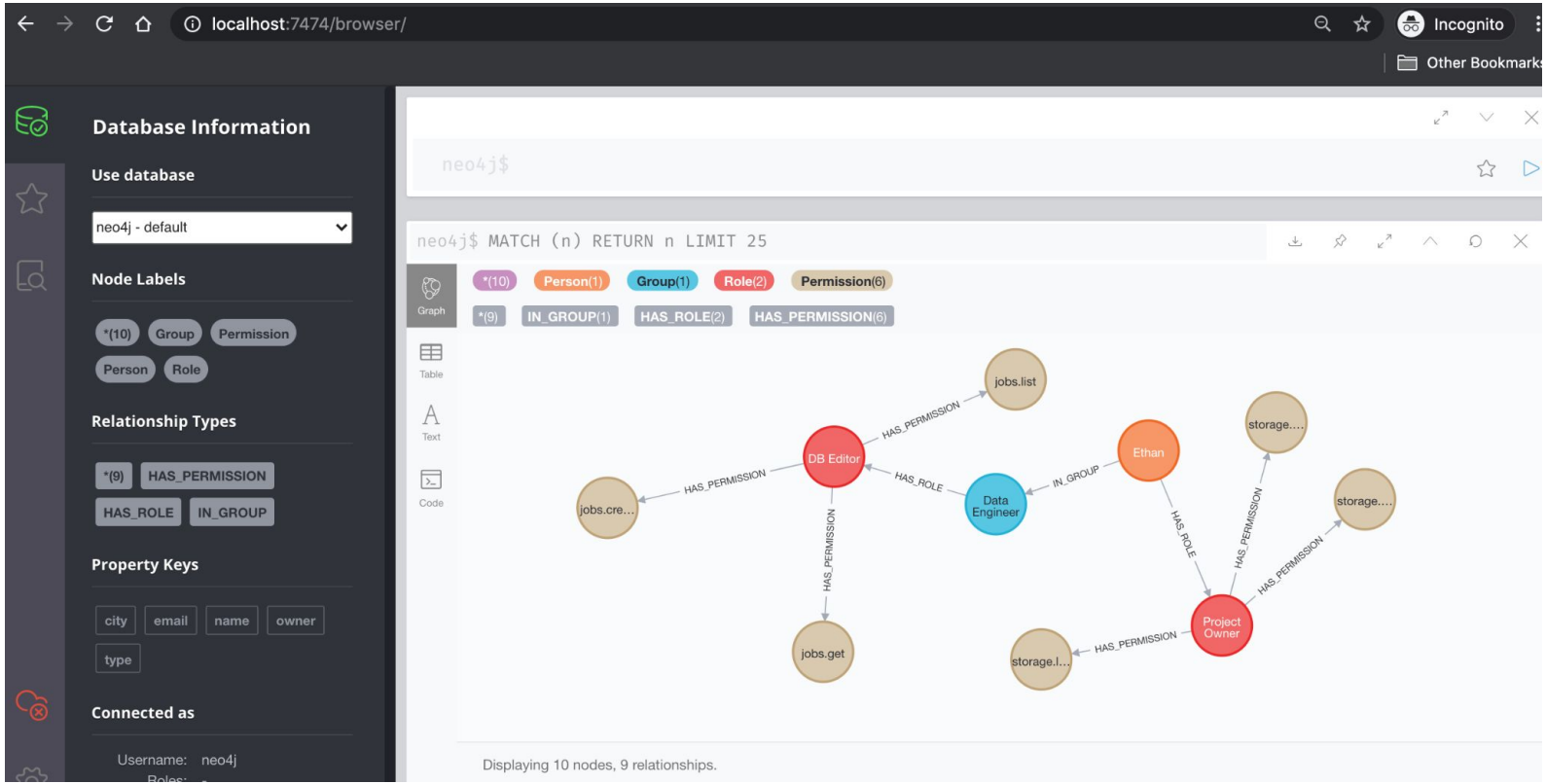
```
MATCH (r:Role {name: "DB Editor"})
MATCH (p:Permission {name: "jobs.list"})
CREATE (r)-[:HAS_PERMISSION]->(p);

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]->(p)
WHERE r.name = "DB Editor"
RETURN r, h, p;
```

# Visualizing the Graph



# Querying the Graph

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

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

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

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

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

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

| 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"|
+-----------------------------------------------------------------+
```

If Ethan had *many more* permissions, we would add a `LIMIT` clause to the end of the query.
If Ethan had *duplicate* permissions, we would use `DISTINCT m` in the `RETURN` clause.

# 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..5]->(m:Permission)
WHERE p.name = "Ethan"
RETURN r, m.name
ORDER BY m;
```

# Updating the Graph

```
MATCH (n:Role {name: "DB Editor"})
MATCH (p:Permission {name: "jobs.create"})
MERGE (n)-[r:HAS_PERMISSION]->(p)
ON MATCH SET r.start_date = "03-26-2021", r.duration = "1_DAY"
RETURN n.name, type(r), r.start_date, r.duration;
```

```
+-----------------------------------------------------------------+
| n.name      | type(r)          | r.start_date | r.duration |
+-----------------------------------------------------------------+
| "DB Editor" | "HAS_PERMISSION" | "03-26-2021" | "1_DAY"    |
+-----------------------------------------------------------------+
```

# Deleting Nodes and Relationships

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

MATCH (p:Person)
DELETE p;


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 Lab

https://github.com/cs327e-spring2021/snippets/wiki/Neo4j-Setup-Guide

https://github.com/cs327e-spring2021/snippets/blob/main/neo4j.ipynb

# Practice Problem

Translate the following scenario into a Cypher query:

**Which persons directed a movie in which they also acted?**

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

**Order the results by person name.**

# Project 6

http://www.cs.utexas.edu/~scohen/projects/Project6.pdf