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

```
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"});
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;

<table>
<thead>
<tr>
<th>p</th>
<th>h</th>
<th>r</th>
</tr>
</thead>
<tbody>
<tr>
<td>(:Person {name: &quot;Ethan&quot;, email: &quot;<a href="mailto:ethan@utexas.edu">ethan@utexas.edu</a>&quot;})</td>
<td>[:IN_GROUP]</td>
<td>(:Group {owner: &quot;Alex&quot;, name: &quot;Data Engineer&quot;})</td>
</tr>
<tr>
<td>(:Person {name: &quot;Ethan&quot;, email: &quot;<a href="mailto:ethan@utexas.edu">ethan@utexas.edu</a>&quot;})</td>
<td>[:HAS_ROLE]</td>
<td>(:Role {name: &quot;Owner&quot;, resource: &quot;Project&quot;})</td>
</tr>
<tr>
<td>(:Group {owner: &quot;Alex&quot;, name: &quot;Data Engineer&quot;})</td>
<td>[:HAS_ROLE]</td>
<td>(:Role {name: &quot;DB Editor&quot;, resource: &quot;Cloud SQL&quot;})</td>
</tr>
</tbody>
</table>
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;

<table>
<thead>
<tr>
<th>r</th>
<th>h</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>{Role {name: &quot;Owner&quot;, resource: &quot;Project&quot;}}</td>
<td>[:HAS_PERMISSION]</td>
<td>{Permission {name: &quot;storage.delete&quot;}}</td>
</tr>
<tr>
<td>{Role {name: &quot;Owner&quot;, resource: &quot;Project&quot;}}</td>
<td>[:HAS_PERMISSION]</td>
<td>{Permission {name: &quot;storage.create&quot;}}</td>
</tr>
<tr>
<td>{Role {name: &quot;Owner&quot;, resource: &quot;Project&quot;}}</td>
<td>[:HAS_PERMISSION]</td>
<td>{Permission {name: &quot;storage.list&quot;}}</td>
</tr>
</tbody>
</table>
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;
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);
```
Querying the Graph

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

<table>
<thead>
<tr>
<th>r</th>
<th>m.name</th>
</tr>
</thead>
<tbody>
<tr>
<td>{::IN_GROUP}, {::HAS_ROLE}, {::HAS_PERMISSION}</td>
<td>&quot;jobs.list&quot;</td>
</tr>
<tr>
<td>{::IN_GROUP}, {::HAS_ROLE}, {::HAS_PERMISSION}</td>
<td>&quot;jobs.get&quot;</td>
</tr>
<tr>
<td>{::IN_GROUP}, {::HAS_ROLE}, {::HAS_PERMISSION}</td>
<td>&quot;jobs.create&quot;</td>
</tr>
<tr>
<td>{::HAS_ROLE}, {::HAS_PERMISSION}</td>
<td>&quot;storage.list&quot;</td>
</tr>
<tr>
<td>{::HAS_ROLE}, {::HAS_PERMISSION}</td>
<td>&quot;storage.create&quot;</td>
</tr>
<tr>
<td>{::HAS_ROLE}, {::HAS_PERMISSION}</td>
<td>&quot;storage.delete&quot;</td>
</tr>
</tbody>
</table>

5. \( \text{MATCH } (\text{p:Person})-[r*]->(\text{m:Permission}) \)
6. \( \text{WHERE } \text{p.name} = "\text{Ethan}" \)
7. \( \text{WITH } \text{distinct } \text{m.name} \text{ as } \text{distinct_perms} \)
8. \( \text{RETURN } \text{distinct_perms} \)
9. \( \text{ORDER BY } \text{distinct_perms}; \)
Querying the Graph

12 MATCH (p:Person)-[r*1..2]-(m:Permission)
13 WHERE p.name = "Ethan"
14 RETURN r, m.name
15 ORDER BY m;

<table>
<thead>
<tr>
<th>r</th>
<th>m.name</th>
</tr>
</thead>
<tbody>
<tr>
<td>[[:HAS_ROLE], [[:HAS_PERMISSION]]]</td>
<td>&quot;storage.list&quot;</td>
</tr>
<tr>
<td>[[:HAS_ROLE], [[:HAS_PERMISSION]]]</td>
<td>&quot;storage.create&quot;</td>
</tr>
<tr>
<td>[[:HAS_ROLE], [[:HAS_PERMISSION]]]</td>
<td>&quot;storage.delete&quot;</td>
</tr>
</tbody>
</table>
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;

Adding node labels:

MATCH (n {name: "Ethan"})
SET n:Principal
RETURN n.name, labels(n) AS labels;
Adding and updating relationship properties:

MATCH (n:Role {name: "DB Editor"})
MATCH (p:Permission {name: "jobs.create"})
MERGE (n)-[: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;

"Renaming" a relationship type:

MATCH (n:Role)-[:HAS_PERMISSION]->(p:Permission)
MERGE (n)-[:HAS_IAM_PERMISSION]->(p)
DELETE rel;
MATCH (r:Role)-[:HAS_IAM_PERMISSION]->(p:Permission)
RETURN r, h, p;
Deleting Relationships and Nodes

Drop the relationships connected to nodes labeled Person:

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

Drop nodes labeled Person:

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

Drop all the nodes and relationships in the current database:

```sql
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