Announcements

• Review session for Test 2
• Test 2 details

Exam rules:
• Open-note and open-book
• Piazza will be disabled during exam
• May not consult with any human in any form
Why Neo4j?

- Designed for storing and querying graphs
- Labeled property graph data model
- Optional schema
- Declarative, SQL-inspired query language (Cypher)
- Rich plugin and extension language (similar to Postgres)
- Open-source, sponsored by Neo4j Inc.
- ACID-compliant transactions
- Clustering option for scaling reads
- Visualization tools (Neo4j Browser, Bloom)
- Optimized for graph traversals
Labeled Property Graph Model

- Person
  - name: Ethan
  - HAS_ROLE ➝ Role
  - HAS_PERMISSION ➝ Permission
  - HAS_PERMISSION ➝ jobs.create

- Role
  - name: DB Viewer
  - HAS_ROLE ➝ Group
  - HAS_PERMISSION ➝ Permission

- Permission
  - name: jobs.create
  - HAS_PERMISSION ➝ Person

- Group
  - name: Data Engineer
  - HAS_PERMISSION ➝ Person
  - HAS_ROLE ➝ Role
Creating Nodes

CREATE ();
CREATE (:Person);

CREATE (:Person {name: "Ethan", email: "ethan@utexas.edu"});
CREATE (:Role {name: "DB Viewer"});
CREATE (:Role {name: "DB Editor"});
CREATE (:Group {name: "Data Engineer"});

CREATE (:Permission {name: "jobs.list"});
CREATE (:Permission {name: "jobs.get"});
CREATE (:Permission {name: "jobs.create"});
Creating Nodes and Relationships

CREATE (:Person)-[r:HAS_ROLE]->(:Role);

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

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

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

MATCH (p:Person {name: "Ethan"})
MATCH (m:Permission {name: "jobs.list"})
CREATE (p)-[:HAS_PERMISSION]->(m);

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

MATCH (g:Group {name: "Data Engineer"})
MATCH (m:Permission {name: "jobs.create"})
CREATE (g)-[:HAS_PERMISSION]->(m);
Neo4j Browser

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

Displaying 7 nodes, 6 relationships.
Querying the Graph

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

+-------------------------+----------+
| type(r)                 | COUNT(r) |
+-------------------------+----------+
| "HAS_ROLE"              |   2      |
| "HAS_GROUP"             |   1      |
| "HAS_PERMISSION"        |   3      |

MATCH (m:Permission)
RETURN COUNT(m);

+-------------------+
| COUNT(m)          |
+-------------------+
|   3               |

MATCH ()-[r:HAS_PERMISSION]->()
RETURN COUNT(r);

+-------------------+
| COUNT(r)          |
+-------------------+
|   3               |

Querying the Graph

MATCH (p:Person {name: "Ethan"})-[r]->(m:Permission)
RETURN p, r, m;

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

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

<table>
<thead>
<tr>
<th>p</th>
<th>r</th>
<th>m</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>[:HAS_GROUP], [:HAS_PERMISSION]</td>
<td>(:Permission {name: &quot;jobs.create&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], [:HAS_PERMISSION]</td>
<td>(:Permission {name: &quot;jobs.get&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_PERMISSION]</td>
<td>(:Permission {name: &quot;jobs.list&quot;})</td>
</tr>
</tbody>
</table>

3 rows available after 53 ms, consumed after another 1 ms
MATCH (r:Role {name: "DB Editor"})
MATCH (p:Permission {name: "jobs.create"})
CREATE (r)-[:HAS_PERMISSION]->(p);

MATCH (r:Role {name: "DB Editor"})
MATCH (p:Permission {name: "jobs.create"})
MERGE (r)-[rel:HAS_PERMISSION]->(p)
ON MATCH SET rel.name = "10-16-2020"
RETURN type(rel), rel.name;

+--------------------------+-----------------+
| type(rel)                | rel.name        |
+--------------------------+-----------------+
| "HAS_PERMISSION"         | "10-16-2020"    |
+--------------------------+-----------------+

1 row available after 1 ms, consumed after another 2 ms
Set 1 properties
Querying the Graph

MATCH (p:Person {name: "Ethan"})-[r*]->(m:Permission)
RETURN m ORDER BY m.name;

MATCH (p:Person {name: "Ethan"})-[r*]->(m:Permission)
RETURN DISTINCT m ORDER BY m.name;
Deleting the Graph

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

MATCH (p:Person)
DELETE p;

MATCH ()-[r]->(m:Permission)
DELETE r;

MATCH (m:Permission)
DELETE m;

MATCH (n)
DETACH DELETE n;
Set up Neo4j

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