NoSQL Graph Databases

Problem Set #4 is graded
Problem Set #5 is due tonight
Graph Databases: Concept

❖ To store entities and relationships between them
  ▪ Nodes are instances of objects
  ▪ Nodes have properties, e.g., name
  ▪ Edges connect nodes and are directed
  ▪ Edges have types (e.g., likes, friend, …)

❖ Nodes are organized by relationships
  ▪ Allow to find interesting patterns
  ▪ example: Get all nodes that are “employee” of “Big Company” and that “likes” “NoSQL Distilled”
Graph Databases: Example

(source: Sadalage & Fowler: NoSQL Distilled, 2012)
Graph Databases: Representatives

- Neo4j
- OrientDB
- TITAN
- Apache Giraph
- InfiniteGraph
Neo4j: An exemplar Graph database

- **Open source** graph database
  - The most **popular**
- **Initial release**: 2007
- **Written in**: Java
- **OS**: cross-platform
- Stores data as **nodes** connected by directed, typed **relationships**
  - With properties on both
  - Called the “property graph”
Neo4j: Data Model

- Fundamental units: **nodes + relationships**
- Both can contain **properties**
  - **Key-value** pairs
  - Value can be of primitive type or an array of primitive type
  - **null** is **not a valid** property value
    - Nulls can be modelled by the absence of a key
Data Model: Relationships

- Directed relationships (edges)
  - Incoming and outgoing **edge**
    - Equally **efficient traversal** in both directions
    - Direction can be **ignored** if not needed by the application
  - Always a **start** and an **end node**
    - Can be recursive
Data Model: Properties

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>true/false</td>
</tr>
<tr>
<td>byte</td>
<td>8-bit integer</td>
</tr>
<tr>
<td>short</td>
<td>16-bit integer</td>
</tr>
<tr>
<td>int</td>
<td>32-bit integer</td>
</tr>
<tr>
<td>long</td>
<td>64-bit integer</td>
</tr>
<tr>
<td>float</td>
<td>32-bit IEEE 754 floating-point number</td>
</tr>
<tr>
<td>double</td>
<td>64-bit IEEE 754 floating-point number</td>
</tr>
<tr>
<td>char</td>
<td>16-bit unsigned integers representing Unicode characters</td>
</tr>
<tr>
<td>String</td>
<td>sequence of Unicode characters</td>
</tr>
</tbody>
</table>
### Examples

<table>
<thead>
<tr>
<th>What</th>
<th>How</th>
</tr>
</thead>
<tbody>
<tr>
<td>get who a person follows</td>
<td>outgoing <em>follows</em> relationships, depth one</td>
</tr>
<tr>
<td>get the followers of a person</td>
<td>incoming <em>follows</em> relationships, depth one</td>
</tr>
<tr>
<td>get who a person blocks</td>
<td>outgoing <em>blocks</em> relationships, depth one</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>What</th>
<th>How</th>
</tr>
</thead>
<tbody>
<tr>
<td>get the full path of a file</td>
<td>incoming <em>file</em> relationships</td>
</tr>
<tr>
<td>get all paths for a file</td>
<td>incoming <em>file</em> and <em>symbolic link</em> relations</td>
</tr>
<tr>
<td>get all files in a directory</td>
<td>outgoing <em>file</em> and <em>symbolic link</em> relations, depth one</td>
</tr>
<tr>
<td>get all files in a directory, excluding symbolic links</td>
<td>outgoing <em>file</em> relationships, depth one</td>
</tr>
<tr>
<td>get all files in a directory, recursively</td>
<td>outgoing <em>file</em> and <em>symbolic link</em> relations</td>
</tr>
</tbody>
</table>

![Diagram](image)
Native Java Interface: Example

Node alice = graphDb.createNode();
alice.setProperty("name", "Alice");
Node bonnie = graphDb.createNode();
bonnie.setProperty("name", "Bonnie");

Relationship a2b = alice.createRelationshipTo(bonnie, FRIEND);
Relationship b2a = bonnie.createRelationshipTo(alice, FRIEND);

a2b.setProperty("quality", "a good one");
b2a.setProperty("since", 2003);

❖ **Undirected edge:**
   - Relationship between the nodes in **both directions**
   - **INCOMING** and **OUTGOING** relationships from a node
Data Model: Traversal + Path

- **Path** = one or more nodes + connecting relationships
  - Typically retrieved as a result of a query or a traversal

- **Traversing a graph** = visiting its nodes, following relationships according to some rules
  - Typically, a subgraph is visited
  - Neo4j: Traversal framework + Java API, Cypher, Gremlin
Traversal Framework

A traversal is influenced by

- **Starting node(s)** where the traversal will begin
- **Expanders** – defines what edges there are to traverse
  - i.e., relationship direction and type
- **Order** – depth-first / breadth-first
- **Uniqueness** – visit nodes (relationships, paths) only once
- **Evaluator** – what to return and whether to stop or continue traversal beyond a current position

Traversal = TraversalDescription + starting node(s)
Traversals Framework – Java API

- `org.neo4j...TraversalDescription`
  - The main interface for defining traversals
    - Can specify branch ordering `breadthFirst()` / `depthFirst()`

- `.relationships()`
  - Adds the relationship type to traverse
    - e.g., traverse only edge types: FRIEND, RELATIVE
    - Empty (default) = traverse all relationships
  - Can also specify direction
    - `Direction.BOTH`
    - `Direction.INCOMING`
    - `Direction.OUTGOING`
Traversals Framework – Java API (3)

❖ org.neo4j...Uniqueness
   ▪ Can be supplied to the TraversalDescription
   ▪ Indicates under what circumstances a traversal may revisit the same position in the graph

❖ Traverser
   ▪ Starts actual traversal given a TraversalDescription and starting node(s)
   ▪ Returns an iterator over “steps” in the traversal
     • Steps can be: Path (default), Node, Relationship
   ▪ The graph is actually traversed “lazily” (on request)
Example of Traversal

TraversalDescription desc =
    db.traversalDescription()
    .depthFirst()
    .relationships(Rels.KNOWS, Direction.BOTH)
    .evaluator(Evaluators.toDepth(3));

// node is 'Ed' (Node[2])
for (Node n : desc.traverse(node).nodes()) {
    output += n.getProperty("name") + ", ";
}

Output: Ed, Lars, Lisa, Dirk, Peter,
Graph Database Summary

❖ Graph databases excel when objects are "indirectly" related to each other. Friends of friends, Cousins, your boss's boss's boss.

❖ Graph databases are suited for finding "structural patterns" in data.
  ▪ If "X" buys "A", "B", "C" are they likely to buy "D"?

❖ When entites and their relationships are clustered
Final Exam: 11/19 from 12pm-3pm
I will be available on Zoom, but you can leave if you want.
Open book, open notes, open-internet
No human communication

15 questions Jupyter Notebook
10 covering materials since the last midterm;
5 comprehensive
Grading Status

- Midterm
  - To my knowledge all issues are resolved and exams are graded
- Problem Sets (lowest score is dropped)
  - Problem Set #5 graded soon!
  - All issues with other problem sets are resolved
- Exercises
  - Everyone will get 100%
- If you still have any issues see me after class today or during my office hours tomorrow
Don't Mess Up!

1. Fill in your signature correctly!

2. Make sure you are logged in when you submit!
(Your cookie lasts for more than 3 hours, so you should logout and then back in just before the exam)

3. Don't submit the empty copy of the exam that you downloaded!

4. Use a local copy of Jupyter if possible.
Summary and What to study

- Relational Model
- Out-of-core sorting
- Normal Forms

- Structured Query Language
- Integrating Databases & programs
- NoSQL
  - BASE, MapReduce, Hadoop
  - Document Model

- Database Indexing
- Query Evaluations
- Query Optimization
- Transactions and Concurrency