Neo4j
A Graph Database Intro

Team members: Jiepeng Zhang, Zhenhua Li & Sha Liu
1. Introduction to Neo4j
   - What is Graph Database?
   - What is Neo4j?
   - Why Neo4j?
   - How to use Neo4j?

2. Proposed Application

3. Application Example
1 Introduction to Neo4j
1.1 What is Graph Database?

Graphs Everywhere…
A graph database?

No: not for storing charts & graphs, or vector artwork

Yes: for storing data that is structured as a graph
✓ remember linked lists, tree?
✓ graphs are the generalized connected data structure
1.1 What is Graph Database?

- You know relational
- Now consider relationships…
1.1 What is Graph Database?

\[ G = (V, E) \]

- A (Graph) – records data in – (Nodes)
- A (Graph) – records data in – (Relationships)
- (Nodes) are – organized by – (Relationships)
- (Nodes & Relationships) – have – (Properties)
Query a graph with a traversal

- A (Traversal) – navigates – a (Graph)
- A (Traversal) – identifies – (Paths)
- (Paths) – order – (Nodes)
Indexes look-up Nodes or Relationships

- An (Index) is a special (Traversal)
- An (Index) maps from properties to either (Nodes or Relationships)
1.1 What is Graph Database?

- We’re talking about a Property Graph

Properties (each a key + value)
A graph database...

- Optimized for the connections between records
- Really, really fast at querying across records
- A database: transactional with the usual operations
- "A relational database may tell you the average age of everyone in this session,
- but a graph database will tell you who is most likely to buy you a beer."
1.2 What is Neo4j?

- A robust and high-performance graph database
- Full enterprise deployment or lightweight projects

- True ACID transactions (Atomicity, Consistency, Isolation, Durability)
- High availability
- Scales to billions of nodes and relationships
- High speed querying through traversals
1.2 Why Neo4j?

What are graphs good for?
A: highly – connected data

- Recommendations
- Business intelligence
- Social computing
- Web analytics
- Geospatial
- And much more!
Example: Social Network “path exists”

- A sample social graph with 1000 persons
- Average 50 friends per person
- `pathExists(a, b)` limited to depth 4
- Caches warmed up to eliminate disk I/O

<table>
<thead>
<tr>
<th></th>
<th># persons</th>
<th>Query time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relational database</td>
<td>1,000</td>
<td>2000ms</td>
</tr>
<tr>
<td>Neo4j</td>
<td>1,000</td>
<td>2ms</td>
</tr>
<tr>
<td>Neo4j</td>
<td>1,000,000</td>
<td>2ms</td>
</tr>
</tbody>
</table>
What's a Graph Database?

Defined as: A graph database is a database that uses graph structures with nodes, edges, and properties to represent and store information.

Manages: Graph

Is a: Neo4j, the world's leading graph database

Records data in: Nodes

Connect: Relationships

Have: Properties

Order: Paths

Identifies: Traversal

Navigates: Index

Maps from: Properties
1.3 Neo4j Details...

- Nodes
- Relationships
- Properties
- Paths
- Traversal
- Server
- Language
Nodes

- The fundamental units that form a graph are nodes and relationships.
- Nodes are often used to represent entities.

Eg. A single node with one property
A relationship connects two nodes, and is always directed, can be viewed as outgoing or incoming relative to a node.

Relationships allow for finding related data.
Properties

- Both nodes and relationships can have properties.
- Properties are key-value pairs where the key is a string. Property values can be either a primitive or an array of one primitive type.

```plaintext
message = 'Hello, '

KNOWS
message = 'brave Neo4j'

message = 'World!'```
Paths

- A path is one or more nodes with connecting relationships, typically retrieved as a query or traversal result.
  
  Eg. A path of length one visiting its nodes, following relationships according to some rules.
  
  \[\text{Depth First / Breadth First}\]

Traversals
Show me some code, please

```java
GraphDatabaseService graphDb = new EmbeddedGraphDatabase("var/neo4j");
Transaction tx = graphDb.beginTx();
try {
    Node steve = graphDb.createNode();
    Node michael = graphDb.createNode();

    steve.setProperty("name", "Steve Vinoski");
    michael.setProperty("name", "Michael Hunger");

    Relationship presentedWith = steve.createRelationshipTo(
        michael, PresentationTypes.PRESENTED_WITH);
    presentedWith.setProperty("date", today);
    tx.success();
} finally {
    tx.finish();
}
```
1.4 How you query this “graph” database?
Gremlin – graph scripting
  ◦ Groovy based Graph Traversal Language
  ◦ Send Gremlin scripts to the Neo4j Server
  ◦ Scripts are executed on the server database
  ◦ Results are returned as Neo4j Node and Relationship representations.
Try it out:

- the graph itself
  - `g.v(0) – node 0`
  - `g.v(0).in – nodes connected to Node 0`
  - `g.v(0).in.name – the names of those Nodes`
  - `g.v(l).outE{it.label == “KNOWS”} – the outgoing “KNOWS”`
  - `g.v(l).outE{it.label == “KNOWS”}.inV.name – knows who?`

Details:
https://github.com/tinkerpop/gremlin/
1.4 How you query this “graph” database?

- Cypher – SQL-like querying
  - WHERE and ORDER BY
- Pattern-matching
- Focus on the clarity of expressing what to retrieve from a graph, not how to do it
**Clauses**

- **START**: Starting points in the graph, obtained via index lookups or by element IDs.
- **MATCH**: The graph pattern to match, bound to the starting points in START.
- **WHERE**: Filtering criteria.
- **RETURN**: What to return.
- **CREATE**: Creates nodes and relationships.
- **DELETE**: Removes nodes, relationships and properties.
- **SET**: Set values to properties.
- **FOREACH**: Performs updating actions once per element in a list.
- **WITH**: Divides a query into multiple, distinct parts.
Create Graph with Cypher

Demo:  http://console.neo4j.org/

```
START john=node:node_auto_index(name = 'John')
MATCH john-[:_friend]->()-[:_friend]->fof
RETURN john, fof
```

<table>
<thead>
<tr>
<th>john</th>
<th>fof</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node[4]{name:&quot;John&quot;}</td>
<td>Node[2]{name:&quot;Maria&quot;}</td>
</tr>
<tr>
<td>Node[4]{name:&quot;John&quot;}</td>
<td>Node[3]{name:&quot;Steve&quot;}</td>
</tr>
</tbody>
</table>

2 rows
4 ms
1.5 Neo4j Server

- Server
  - http://localhost:7474/webadmin
• Embedded in application
• REST API
  • HTTP protocol and JSON
  • Insert, delete and access data easily from any programming language

• Ex. Create the nodes
  • Send a HTTP request containing a JSON payload to the server
  • The server will have created a new node in the database
  • Responded with a 201 Created response and a Location header with the URI of the newly created node
1.6 Language

Java, JavaScript, Node.js, Python, Rails, PHP, .NET

- Embedded in application
- REST API

<table>
<thead>
<tr>
<th>name</th>
<th>language/framework</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neo4j.rb</td>
<td>JRuby</td>
<td><a href="https://github.com/andreasronge/neo4j">https://github.com/andreasronge/neo4j</a></td>
</tr>
<tr>
<td>Neo4jDjango</td>
<td>Python, Django</td>
<td><a href="https://github.com/scholrly/neo4django">https://github.com/scholrly/neo4django</a></td>
</tr>
<tr>
<td>Neo4js</td>
<td>JavaScript</td>
<td><a href="https://github.com/neo4j/neo4js">https://github.com/neo4j/neo4js</a></td>
</tr>
</tbody>
</table>
| Gremlin       | Java, Groovy       | Section 18.18. “Gremlin Plugin”,
                                      | https://github.com/tinkerpop/gremlin/wiki |
| Neo4j-Scala   | Scala              | https://github.com/FaKod/neo4j-scala     |
| Borneo        | Clojure            | https://github.com/wagjo/borneo          |
More Information...

- Documentation
  - docs.neo4j.org-tutorials + reference
  - Neo4j in Action
  - Good Relationships

- Get Neo4j
  - http://www.neo4j.org/download
  - https://addons.heroku.com/neo4j
2 Proposed Application

- A Movie Recommendation System
- Dataset
- The graph database --- Neo4j
- The graph traversal language --- Gremlin
Oracle 10g Programming: A Primer [Paperback]
Rajshakhar Sunderraman (Author)

Buy New
$47.84 Prime

Only 6 left in stock (more on the way).
Ships from and sold by Amazon.com. Gift-wrap available.

Want it tomorrow, April 20? Order within 7 hrs 21 mins, and choose Saturday Delivery at checkout. Details

21 new from $39.63 36 used from $19.95

Earn $5 for each friend you refer to Amazon Student — see details

Customers Who Bought This Item Also Bought
- Introduction to Algorithms $80.00
- Programming the World Wide Web (9th Edition) $80.00
- New Perspectives on Microsoft Access (9th Edition) $104.49
- Database Management Systems $112.99
- Database System Concepts $147.80
Recommender systems or recommendation systems (sometimes replacing "system" with a synonym such as platform or engine) are a subclass of information filtering system that seek to predict the 'rating' or 'preference' that user would give to an item (such as music, books, or movies) or social element (e.g. people or groups) they had not yet considered, using a model built from the characteristics of an item (content-based approaches) or the user's social environment (collaborative filtering approaches). \[1\][2]

Recommender systems have become extremely common in recent years. A few examples of such systems:
2.1 Generating a MovieRatings Graph

- Parse the raw MovieLens data according to the graph schema.
2.2 Traversing the MovieLens Graph

- Recommendation Algorithms
  - Collaborative filtering
    - The rating/liking/preference behavior of users is correlated in order to recommend the favorites of one user to another, similar user.

- Content-based recommendation
  - If a particular item is liked, then its features are analyzed in order to find other items with analogous features.
I like the movies you like, what other movies do you like that I haven’t seen?

- Let start from Toy Story.
- Which users have Toy Story more than 3 stars?
The traversal doesn’t yield useful information. However, when it is used within a larger path expression, collaborative filtering is effected.
Example: Which users gave Toy Story more than 3 stars and what other movies did they give more than 3 stars to?

1. Start from Toy Story
2. Get the incoming rated edges
3. Filter out those edges whose star property is less than 4.
4. Get the tail user vertices of the remaining edges.
5. Get the rating edges of those user vertices
6. Filter out those edges whose star property is less than 4.
7. Get the head movie vertices of the remaining edges
8. Get the string title property of those movie vertices

```
v.inE('rated').filter(it.getProperty('stars') > 3).outV.outE('rated').filter(it.getProperty('stars') > 3).inV.title
```

highly co-rated
These atomic-steps together can be bundled into a user defined step. --- “corated”

gremlin> v.corated(3).title[0..4]

Will return only 5 results.

Given that there are 268,493 highly rated paths from Toy Story to other movies and only 3,353 of those movies are unique, it is possible to use these duplicates as a ranking mechanism—ultimately, a recommendation.
Which movies are most highly co-rated with Toy Story?

gremlin>m=[:]

gremlin> v.corated(3).title.groupCount(m) >> -1

gremlin> m.sort{a,b -> b.value <=> a.value}[0..9]
Mixing in Content– based Recommendation

A traversal that yields Toy Story good movies of the same genre.

Which movies are most highly co-rated with Toy Story that share a genre/all genres with Toy Story?

```gremlin
m = [:]
x = [] as Set
v.out('hasGenera').aggregate(x).back(2).corated(3).filter{it != v}.filter{it.out('hasGenera')>>() as Set == x}.title.groupCount(m) >> -1
===>null
m.sort{a,b -> b.value <=> a.value}[0..9]
```
Yeah! Got Movies Recommendations!
3 Application Example

http://neoflix.herokuapp.com/
Thank You!

Questions?
Neo4j is a robust transactional property graph database. Due to its graph data model, Neo4j is highly agile and blazing fast. For connected data operations, Neo4j runs a thousand times faster than relational databases.

More than 20 of the Global 2000, hundreds of startups and thousands of community members use Neo4j in a wide variety of use cases such as social applications, recommendation engines, fraud detection, resource authorization, network & data center management and much more.