

Joshua Catoe

08/05/18

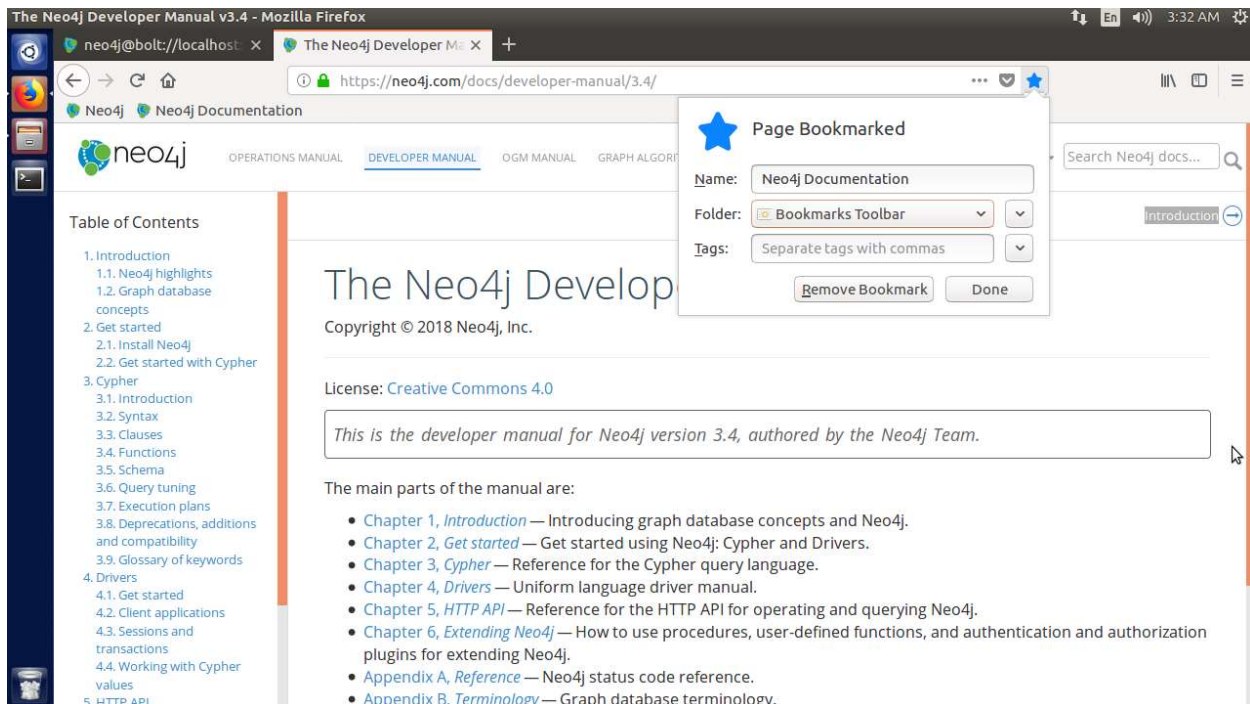
Neo4J

*Note: I installed the neo4j-gremlin functionality with help from Casey in the discussion forum, but I ran into a problem where I could not create a default Neo4j database/node. I believe this was caused by a missing dependency, but I could not find a solution. In some cases where this prevent me from progressing, I have included code that I'm certain would have worked, had I been able to solve the problem.

Day 1

Find:

1. Bookmark the Neo4j wiki.



There is no longer a Neo4j wiki. The official documentation can be found by clicking the book icon in the web interface, followed by the Developer Manual link.

2. Bookmark the Gremlin steps from the wiki or API.

The screenshot shows a web browser window titled "TinkerPop3 Documentation - Mozilla Firefox". The address bar shows the URL "tinkerpop.apache.org/docs/3.1.0-incubating/#neo4j-gremlin". The page is titled "Neo4j-Gremlin". On the left, there is a navigation menu with links to "Range Queries", "Logical Operators", "Traverser Methods", "Utilities Plugin", "Benchmarking and Profiling", "Describe Graph", "Implementations", "Graph System Provider", "Requirements", "Implementing Gremlin-Core", "OLTP Implementations", "OLAP Implementations", "IO Implementations", "Validating with Gremlin-Test", "Accessibility via GremlinPlugin", "In-Depth Implementations", "TinkerGraph-Gremlin", "Configuration", "Neo4j-Gremlin", "Indices", "Multi/Meta-Properties", "Cypher", "Multi-Label", and "Loaders with". The main content area shows the "Neo4j-Gremlin" section. It contains a code block with XML-like dependency tags:

```
<dependency>
  <groupId>org.apache.tinkerpop</groupId>
  <artifactId>neo4j-gremlin</artifactId>
  <version>3.1.0-incubating</version>
</dependency>
<!-- neo4j-tinkerpop-api-impl is NOT Apache 2 licensed - more information below -->
<dependency>
  <groupId>org.neo4j</groupId>
  <artifactId>neo4j-tinkerpop-api-impl</artifactId>
  <version>0.1-2.2</version>
</dependency>
```

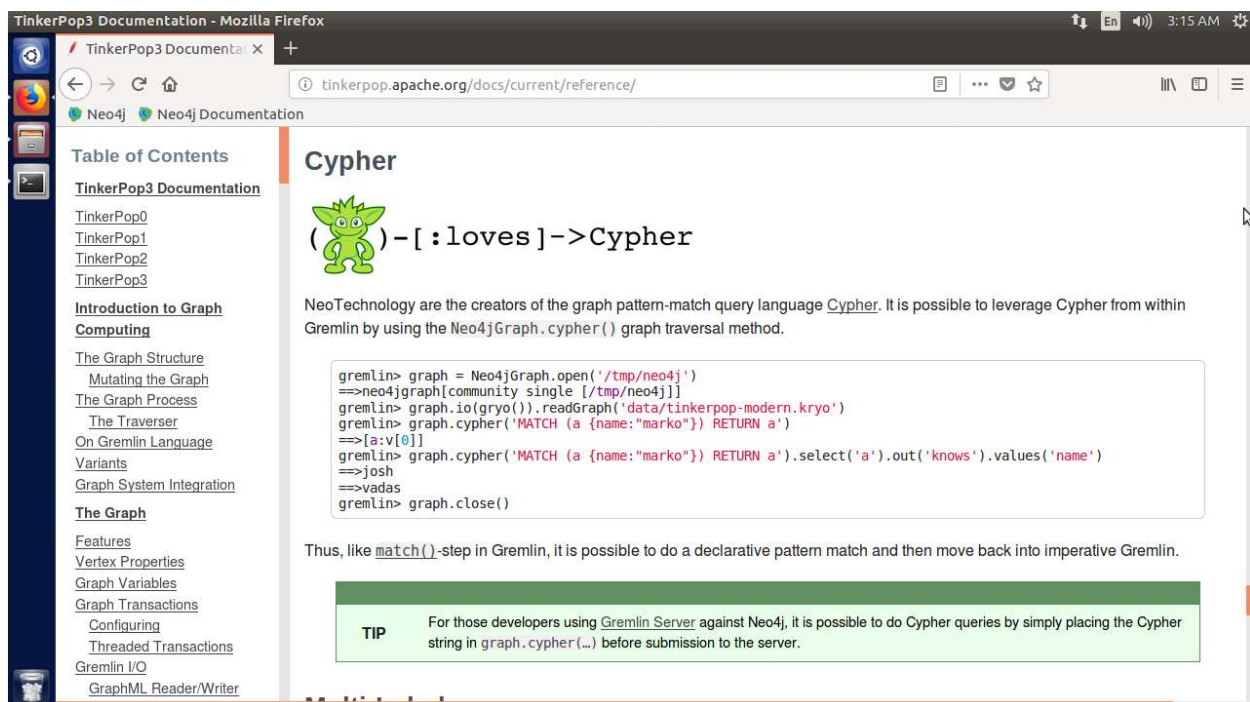
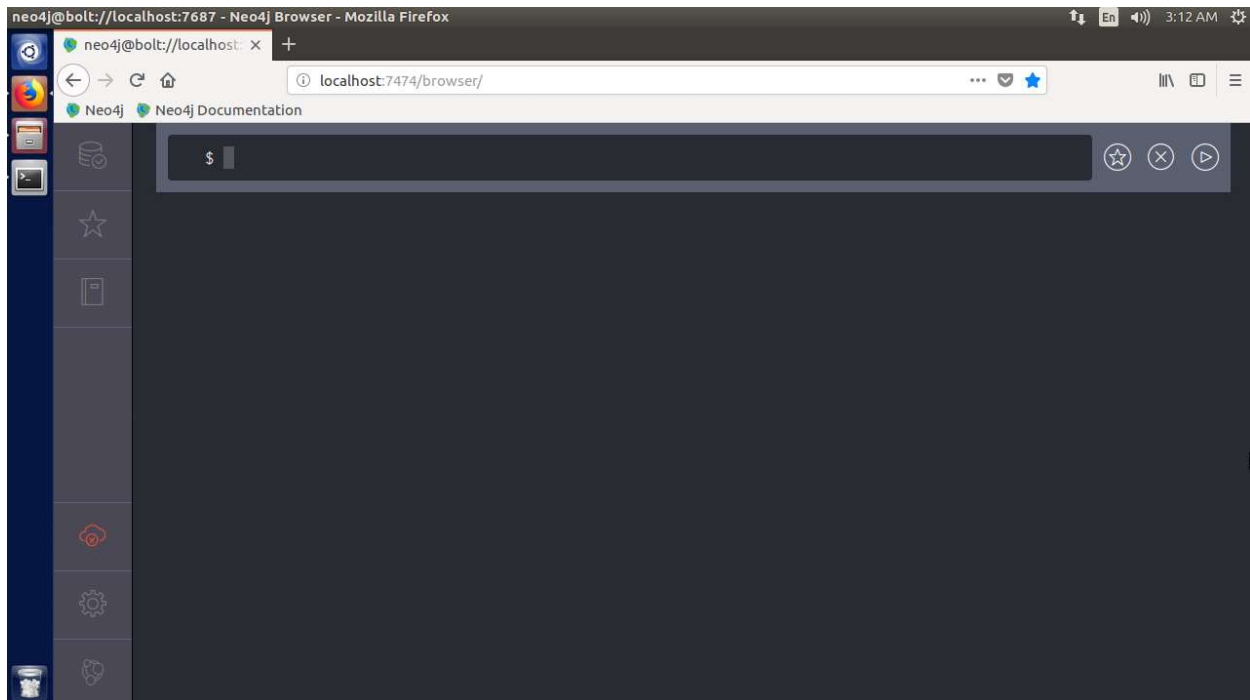
 Below this, it says "Neo Technology are the developers of the OLTP-based Neo4j graph database." There is a "CAUTION" box with text: "Unless under a commercial agreement with Neo Technology, Neo4j is licensed AGPL. The neo4j-gremlin module is licensed Apache2 because it only references the Apache2-licensed Neo4j API (not its implementation). Note that neither the Gremlin Console nor Gremlin Server distribute with the Neo4j implementation binaries. To access the binaries, use the :install command to download binaries from Maven Central Repository." At the bottom, there is a code block with Gremlin console commands:

```
gremlin> :install org.apache.tinkerpop neo4j-gremlin 3.1.0-incubating
=>Loaded: [org.apache.tinkerpop, neo4j-gremlin, 3.1.0-incubating] - restart the console to use [tinkerpop.neo
gremlin> :q
...
gremlin> :plugin use tinkerpop neo4j
```

 The bottom of the browser window shows a search bar with "neo4j" entered and a status bar indicating "2 of 170 matches".

The Neo4j documentation has no instructions for Gremlin, and the web interface has no implementation of it. Instead, you must use the Gremlin Console from Apache and install a plugin for Neo4j. The instructions for doing so can be found in the Tinkerpop 3 documentation under Implementations -> Neo4j-Gremlin.

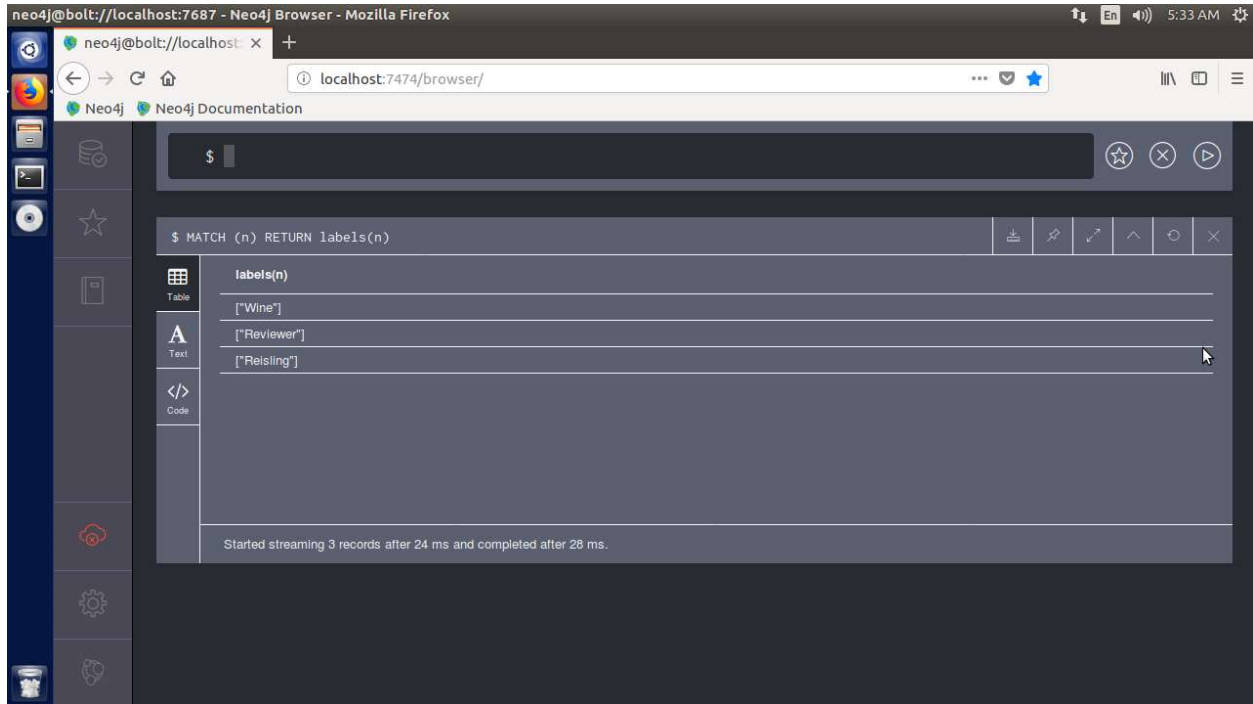
3. Find two other Neo4j shells (such as the Cypher shell in the admin console).



The Cypher shell can be used at <http://localhost:7474/browser>. The only other shell I was able to find was Gremlin. It can be used to run Cypher commands, so I'm assuming that it counts as a Neo4j shell.

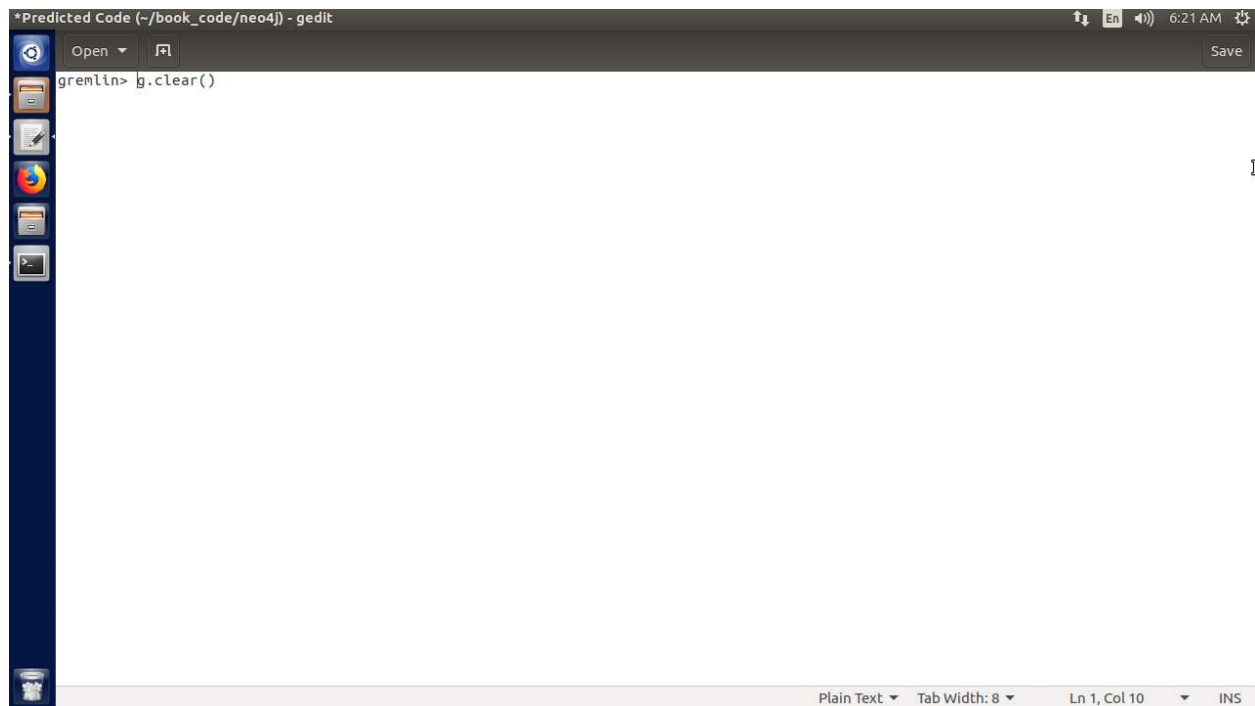
Do:

1. Query all node names with another shell (such as the Cypher query language).



All it takes to query node labels in Cypher is the simple query shown above. **MATCH (n)** to find all nodes and **RETURN labels(n)** to display their label names. The function **labels()** finds all labels for a specific node.

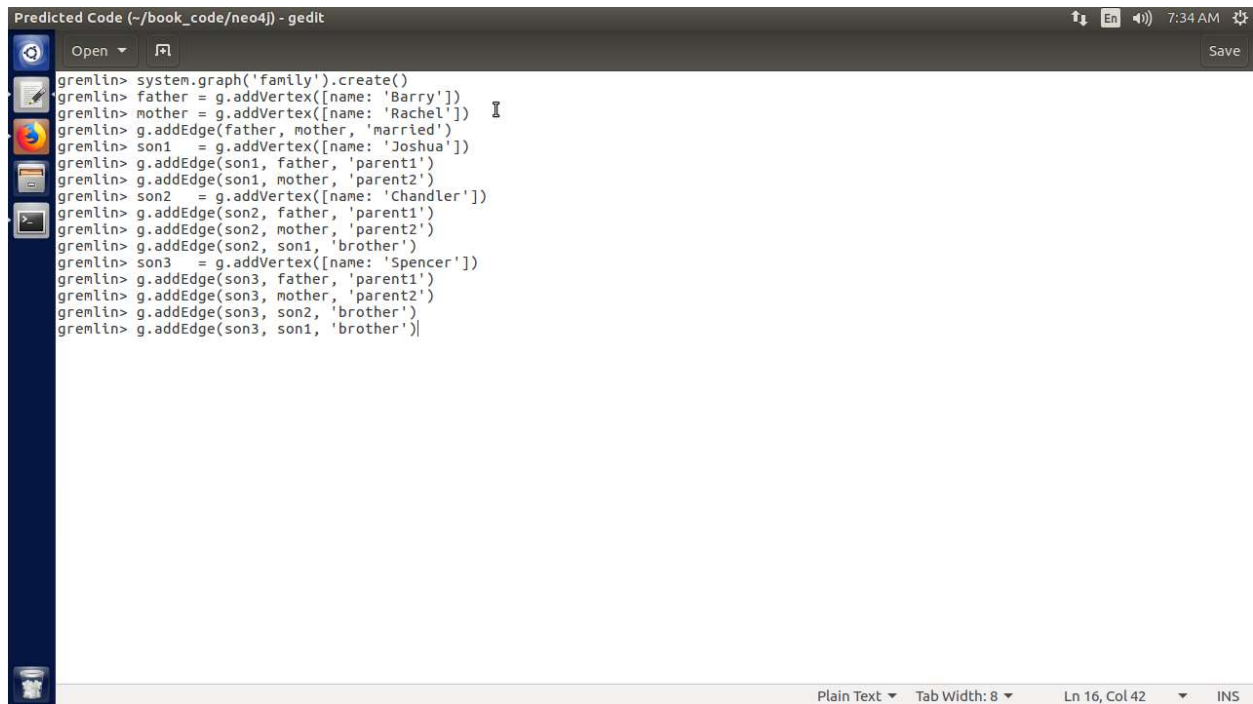
2. Delete all the nodes and edges in your database.



The screenshot shows a gedit text editor window titled '*Predicted Code (~/.book_code/neo4j) - gedit'. The window has a dark theme. On the left side, there is a vertical toolbar with icons for opening files, saving, undo, redo, and other editing functions. The main text area is white and contains a single line of text: 'grenlin> g.clear()'. The status bar at the bottom of the window displays 'Plain Text', 'Tab Width: 8', 'Ln 1, Col 10', and 'INS'. The system tray at the top right shows the time as 6:21 AM and some system icons.

For this, I went with what the book calls the “dangerous” option. **g.clear()** completely clears the graph.

3. Create a new graph that represents your family.

A screenshot of a gedit code editor window. The title bar reads "Predicted Code (~/.book_code/neo4j) - gedit". The editor contains a Gremlin script. The script starts with creating a graph, then adds vertices for Barry (father), Rachel (mother), Joshua (son1), Chandler (son2), and Spencer (son3). It then adds edges: 'married' between father and mother, 'parent1' between father and each son, 'parent2' between mother and each son, and 'brother' between the sons. The status bar at the bottom indicates "Plain Text", "Tab Width: 8", "Ln 16, Col 42", and "INS".

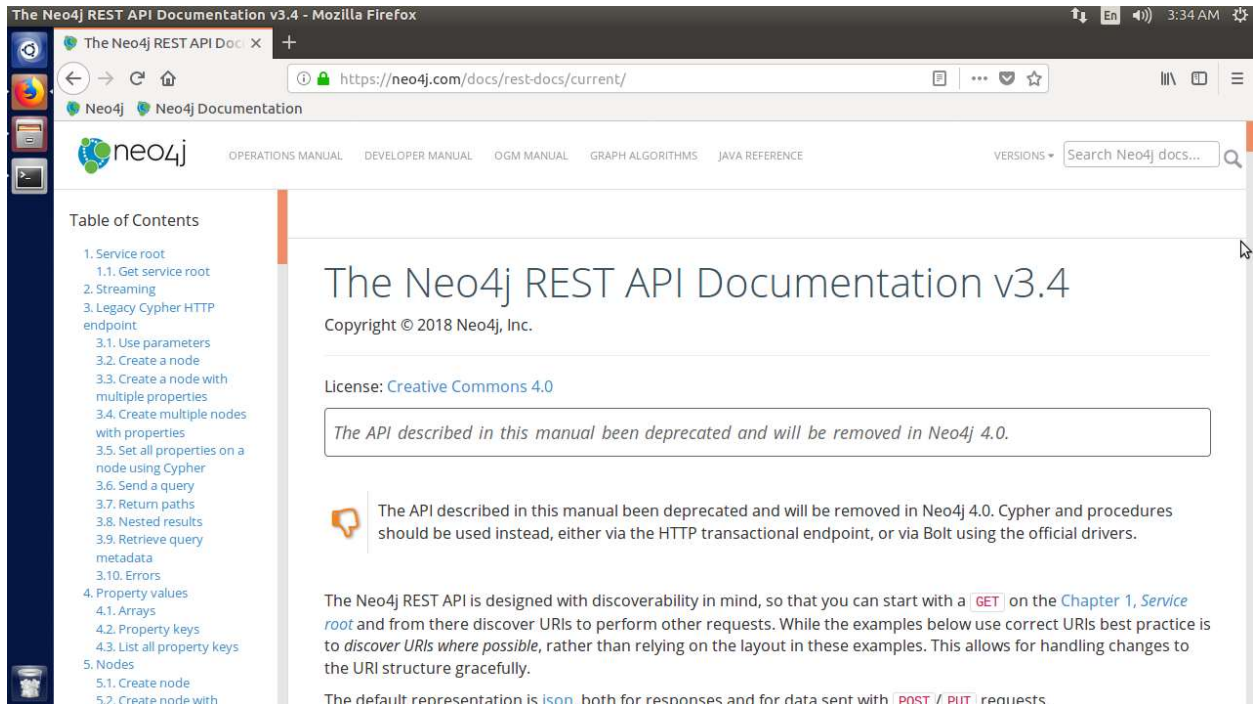
```
gremlin> system.graph('family').create()
gremlin> father = g.addVertex([name: 'Barry'])
gremlin> mother = g.addVertex([name: 'Rachel'])
gremlin> g.addEdge(father, mother, 'married')
gremlin> son1 = g.addVertex([name: 'Joshua'])
gremlin> g.addEdge(son1, father, 'parent1')
gremlin> g.addEdge(son1, mother, 'parent2')
gremlin> son2 = g.addVertex([name: 'Chandler'])
gremlin> g.addEdge(son2, father, 'parent1')
gremlin> g.addEdge(son2, mother, 'parent2')
gremlin> g.addEdge(son2, son1, 'brother')
gremlin> son3 = g.addVertex([name: 'Spencer'])
gremlin> g.addEdge(son3, father, 'parent1')
gremlin> g.addEdge(son3, mother, 'parent2')
gremlin> g.addEdge(son3, son2, 'brother')
gremlin> g.addEdge(son3, son1, 'brother')
```

I used the **system.graph().create()** function to actually create the graph, then used **g.addVertex()** to add the vertexes for the family members. In between vertexes, I added the edges with **g.addEdge()** to represent the relationships (married, parent, brother, etc.).

Day 2

Find:

1. Bookmark the documentation for the Neo4j REST API.



The REST API documentation can be found at <http://neo4j.com/docs/rest-docs/current/>, although it is due to be remove in version 4.0 of Neo4j.

2. Bookmark the API for the JUNG project and the algorithms it implements.

The screenshot shows the JUNG 2.0 API documentation page. The browser window title is "Overview (jung2 2.0 API) - Mozilla Firefox". The address bar shows the URL "jung.sourceforge.net/doc/api/index.html". The page has a navigation bar with links: Overview, Package, Class, Use, Tree, Deprecated, Index, Help. Below the navigation bar, the title "jung2 2.0 API" is displayed. The main content area is titled "Packages" and contains a table with the following data:

Package	Description
edu.uci.ics.jung.algorithms.blockmodel	Support for establishing and maintaining graph element equivalence (such as in blockmodeling).
edu.uci.ics.jung.algorithms.cluster	Mechanisms for identifying clusters in graphs.
edu.uci.ics.jung.algorithms.filters	Filtering mechanisms that produce subgraphs of an original graph.
edu.uci.ics.jung.algorithms.flows	Methods for calculating properties relating to network flows (such as max flow/min cut).
edu.uci.ics.jung.algorithms.generators	Methods for generating new (often random) graphs with various properties.
edu.uci.ics.jung.algorithms.generators.random	Methods for generating random graphs with various properties.
edu.uci.ics.jung.algorithms.importance	
edu.uci.ics.jung.algorithms.layout	Algorithms for assigning 2D coordinates (typically used for graph visualizations) to vertices.
edu.uci.ics.jung.algorithms.layout.util	Utility classes for updating layout positions.
edu.uci.ics.jung.algorithms.layout3d	
edu.uci.ics.jung.algorithms.matrix	Mechanisms for dealing with graphs as matrices.

The left sidebar shows a list of "All Classes" including `AbsoluteCrossoverScalingControl`, `AbstractEdgeShapeTransformer`, `AbstractElementParser`, `AbstractGraph`, `AbstractGraphMouseListener`, `AbstractIterativeScorer`, `AbstractIterativeScorerWithPriors`, `AbstractLayout`, `AbstractLayoutSupport`, `AbstractLayoutSupport.Lens`, `AbstractLayoutSupport.LensControls`, `AbstractMetadata`, `AbstractModalGraphMouseListener`, `AbstractPerspectiveTransformSupport`, `AbstractPerspectiveTransformSupport3D`, `AbstractPickedState`, and `AbstractPopupGraphMouseListener`.

The current JUNG API can be found at <http://jung.sourceforge.net/doc/api/index.html>.

3. Find a binding or REST interface for your favorite programming language.

The screenshot shows the GitHub repository for "libulfius". The browser window title is "GitHub - babelouest/ulfius: HTTP Framework for REST API in C, using JSON or not, with websockets or not, with streaming data or not - Mozilla Firefox". The address bar shows the URL "https://github.com/babelouest/ulfius". The repository page shows a list of files: `LICENSE`, `Makefile`, `README.md`, and `libulfius.pc.in`. The `README.md` file is selected and its content is displayed. The content of the `README.md` file is as follows:

Ulfius

HTTP Framework for REST Applications in C.

Based on [GNU Libmicrohttpd](#) for the backend web server, [Jansson](#) for the json manipulation library, and [Libcurl](#) for the http/smtplib client API.

Used to facilitate creation of web applications in C programs with a small memory footprint, as in embedded systems applications.

You can create webservice in HTTP or HTTPS mode, stream data, or implement server websockets.

Hello World! example application

There is a REST interface for C on GitHub called Ulfius.

Do:

1. Turn the path-finding portion of the Kevin Bacon algorithm into its own step. Then implement a general-purpose Groovy function (for example, `def actor_path(g,name1,name2){...}`) that accepts the graph and two names and compares the distance.
2. Choose and run one of the many JUNG algorithms on a node (or the data set, if the API demands it).

The top screenshot shows a web browser displaying the JUNG API documentation for the `DijkstraShortestPath` class. The browser address bar shows `http://jung.sourceforge.net/doc/api/index.html`. The page title is "Class DijkstraShortestPath<V,E>". The documentation includes a summary, a description of the class, and a list of implemented interfaces.

The bottom screenshot shows a code editor with a Groovy script. The script is a simple test case for the `DijkstraShortestPath` class, using the `gremlin` library to find the shortest path between two actors.

```
gremlin> g.getPath(it.name=='Clint Eastwood', it.name=='Morgan Freeman')
```

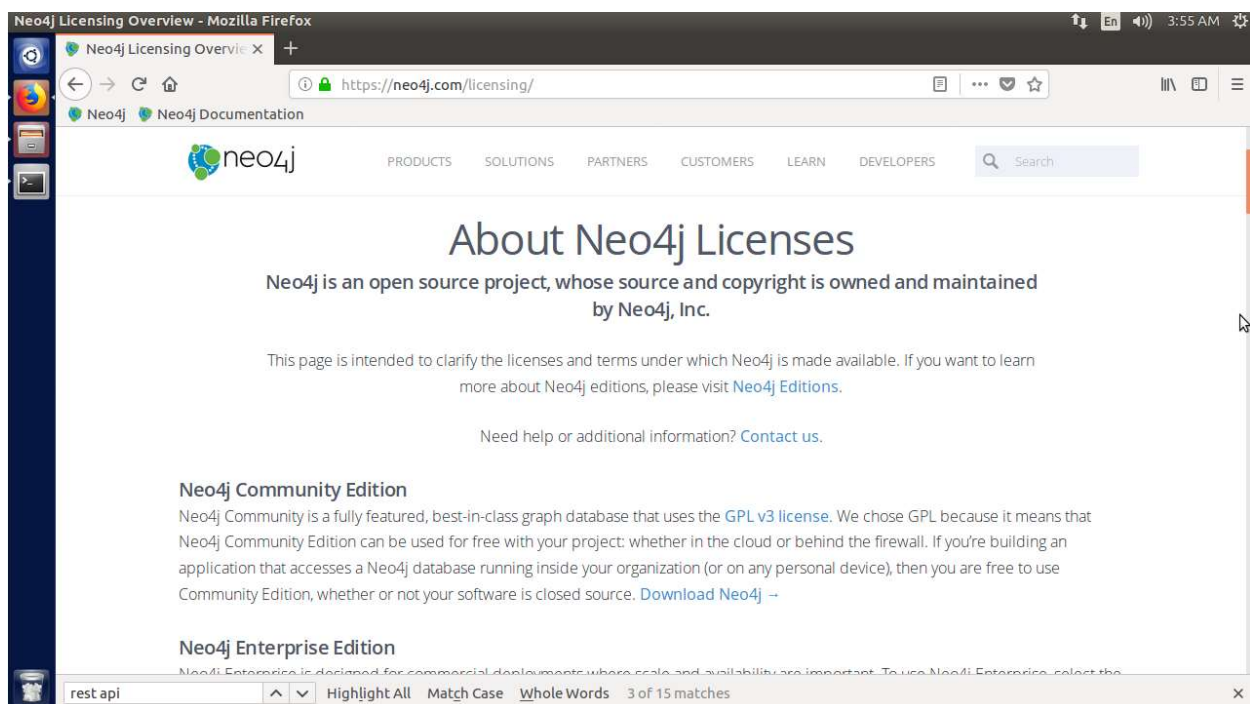
I chose the DijkstraShortestPath algorithm, specifically the **getPath()** method. This takes two vertices as parameters and shortest distance between the two in the graph.

3. Install your driver of choice and use it to manage your company graph with the people and the roles they play, with edges describing their interactions (reports to, works with). If your company is huge, just try your close teams; if you're with a small organization, try including some customers. Find the most well-connected person in the organization by closest distance to all other nodes.

Day 3

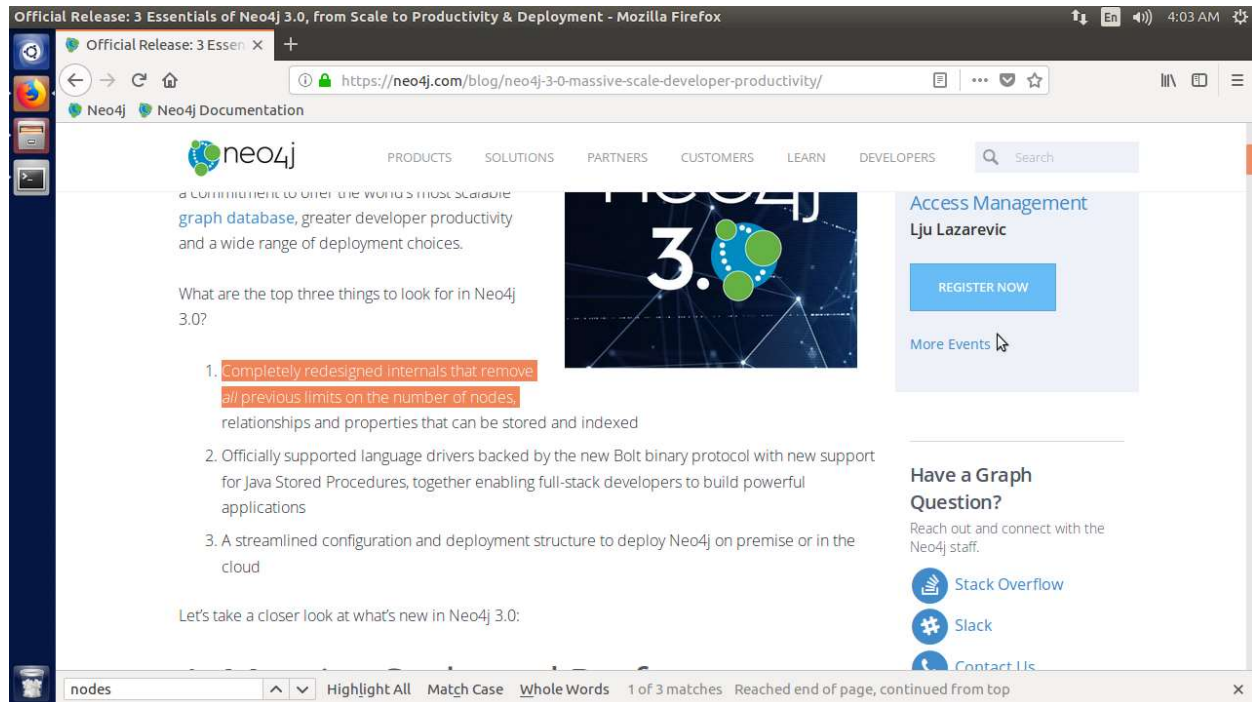
Find:

1. Find the Neo4j licensing guide.



Information about all Neo4j licenses can be found on the Licenses page at <http://neo4j.com/licensing>.

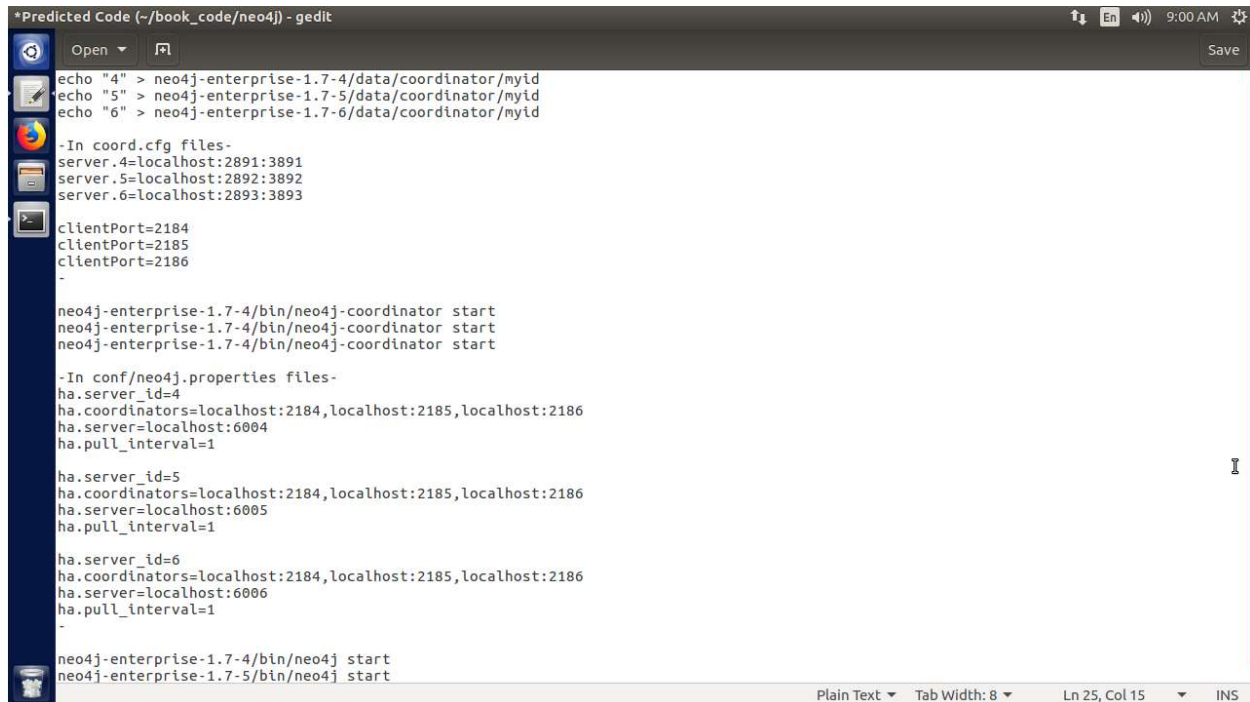
2. Answer the question, “What is the maximum number of nodes supported?”



Detailed in an official Neo4j blog post, as of Neo4j 3.0.0 there are no limits on the number of nodes that can be supported. The previous number was 34 billion (as mentioned in the post).

Do:

1. Replicate Neo4j across three physical servers.



The screenshot shows a gedit editor window titled '*Predicted Code (-/book_code/neo4j) - gedit'. The editor contains configuration files for three Neo4j servers, each with a unique ID (4, 5, and 6). The files are organized into sections: 'coord.cfg files', 'clientPort', 'neo4j-enterprise-1.7-4/bin/neo4j-coordinator start', 'In conf/neo4j.properties files', and 'neo4j-enterprise-1.7-4/bin/neo4j start'. The configuration for each server includes the server ID, coordinators, server address, and pull interval. The status bar at the bottom indicates 'Plain Text', 'Tab Width: 8', 'Ln 25, Col 15', and 'INS'.

```
*Predicted Code (-/book_code/neo4j) - gedit
Open
Save

echo "4" > neo4j-enterprise-1.7-4/data/coordinator/myid
echo "5" > neo4j-enterprise-1.7-5/data/coordinator/myid
echo "6" > neo4j-enterprise-1.7-6/data/coordinator/myid

-In coord.cfg files-
server.4=localhost:2891:3891
server.5=localhost:2892:3892
server.6=localhost:2893:3893

clientPort=2184
clientPort=2185
clientPort=2186
-

neo4j-enterprise-1.7-4/bin/neo4j-coordinator start
neo4j-enterprise-1.7-4/bin/neo4j-coordinator start
neo4j-enterprise-1.7-4/bin/neo4j-coordinator start

-In conf/neo4j.properties files-
ha.server_id=4
ha.coordinators=localhost:2184,localhost:2185,localhost:2186
ha.server=localhost:6004
ha.pull_interval=1

ha.server_id=5
ha.coordinators=localhost:2184,localhost:2185,localhost:2186
ha.server=localhost:6005
ha.pull_interval=1

ha.server_id=6
ha.coordinators=localhost:2184,localhost:2185,localhost:2186
ha.server=localhost:6006
ha.pull_interval=1
-

neo4j-enterprise-1.7-4/bin/neo4j start
neo4j-enterprise-1.7-5/bin/neo4j start
```

Plain Text Tab Width: 8 Ln 25, Col 15 INS

Every Neo4j server must have a different name in a different directory, so I just used 4, 5, and 6. Then I set the ports and client ports for each server in their respective config files and started a coordinator in each server's directory. Finally, under the properties file for each server I added the high availability settings and then started each server from the console.

2. Set up a load balancer using a web server like Apache or Nginx and connect to the cluster using the REST interface. Execute a Gremlin script command.