Neo4j - New Neo4j-import

Neo4j has recently announced the 2.2 Milestone 2 release. Among the exciting features is the improved and fully integrated “Superfast Batch Loader”. This utility (unsurprisingly) called neo4j-import, now supports large scale non-transactional initial loads (of 10M to 10B+ elements) with sustained throughputs around 1M records (node or relationship or property) per second. Neo4j-import is available from the command line on both Windows and Unix.

In this post, I'll walk through how to set up the data files, some command line options and then document the performance for importing a medium size data set.

Data set

The data set that we will use is Medicare Provider Utilization and Payment Data.

The Physician and Other Supplier PUF contains information on utilization, payment (allowed amount and Medicare payment), and submitted charges organized by National Provider Identifier (NPI), Healthcare Common Procedure Coding System (HCPCS) code, and place of service. This PUF is based on information from CMS’s National Claims History Standard Analytic Files. The data in the Physician and Other Supplier PUF covers calendar year 2012 and contains 100% final-action physician/supplier Part B non-institutional line items for the Medicare fee-for-service population.

For the data model, I created doctor nodes, address nodes, procedure nodes and procedure detail nodes.

A (doctor)-[:LOCATED_AT]-(Address)
A (doctor)-[:PERFORMED]-(procedure)
A (procedure) -[:CONTAINS]-(procedure_details)

The model is shown below:

Import Tool Notes

The new neo4j-import tool has the following capabilities:

- Fields default to be comma separated, but a different delimiter can be specified.
- All files must use the same delimiter.
- Multiple data sources can be used for both nodes and relationships.
- A data source can optionally be provided using multiple files.
- A header which provides information on the data fields must be on the first row of each data source.
- Fields without corresponding information in the header will not be read.
- UTF-8 encoding is used.

File Layouts

The header file must have an :ID and can have multiple properties and a :LABEL value. The :ID is a unique value that is used during node creation but more importantly is used during relationships creation. The :ID value is used later as the :START_ID or :END_ID value which are used to create the relationships. :ID values can be integers or strings as long as they are unique.

The :LABEL value is used to create the Label in the Property Graph. A label is a named graph construct that is used to group nodes into sets; all nodes labeled with the same label belongs to the same set.

For example:

In this case, the :ID is the unique number, and the :LABEL column sets the doctor type. Name and NPI are property values added to the node.

The relationships file header contains a :START_ID, :END_ID and can optionally include properties and :TYPE. The :START_ID and the :END_ID link back to previously :ID values from the nodes files. The :TYPE is used to set the relationship type. All relationships in Neo4j have a relationship type.

Separate Header Files

It can be convenient to put the header in a separate file. This makes it easier to edit the header, as you avoid having to open a huge data file to just change the header. To use a separate header file in the command line, see the example below:

Command Line Usage

Under Unix/Linux/OSX, the command is named neo4j-import. Under Windows, the command is named Neo4jImport.bat.

Depending on the installation type, the tool is either available globally, or used by executing ./bin/neo4j-import or bin\Neo4jImport.bat from inside the installation directory.

Additional details can be found online.

Loading Medicare Data using neo4j-import

From the command line, we run the neo4j-import command. The --into option is used to indicate where to store the new database. The directory must not contain an existing database.

The --nodes option indicate the header and nodes files. Multiple --nodes can be used.
The --relationships option indicates the header and relationship files. Multiple --relationships can be used.

Results

Running the load script on my 16GB Macbook Pro with an SSD, I was able to load 20.7M nodes, 23.4M relationships and 35.6M properties in 2 minutes and 5 seconds.

Conclusion

The new neo4j-import tool provides significant performance improvements and ease-of-use for the initial data loading of a Neo4j database. Please download the latest milestone release and give the neo4j-import tool a try.

We are eager to hear your feedback. Please post it to the Neo4j Google Group, or send us a direct email at feedback@neotechnology.com.