Systems Research Documentation – Created By: Alexander Kostoff

How to install neo4j Desktop and create a database:

1. Go to <https://neo4j.com/download/> and download the application.
2. Launch the installer, choose if you want it only installed on your account, then click next, and install.
3. You will be presented with the Neo4j Primer Project already created. To create a local database click Add Database, then click on Create a local database.
4. Set the password to “test”, this is because the python programs wrote have test in the password for connecting.
5. Then click create.
6. Make sure to start the database before using the python programs.

How to connect to the neo4j database from the python programs. These steps are the same in Linux as well.

In the python programs there is code: 

This calls a class initializer function of the neo4j class used that will run code to connect to and manipulate the database. In this, is the database URL, username, and password. Neo4j is the default username and the password is test. The information in each python program will need to be changed if a different URL, username, and/or password are used.

The class initializer:



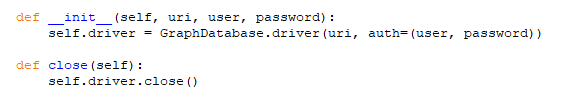
Make sure neo4j is installed in python:

1. This can be done with pip install neo4j.
2. Then in the program have:

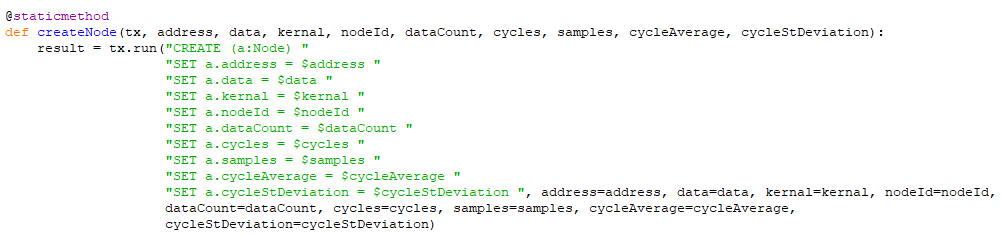
The Research.py program takes in data from a file that contains the processes of an application and inserts them into a tree. From that tree the nodes are inserted into a graph database, neo4j.

The neo4j class integrates python with the neo4j database:

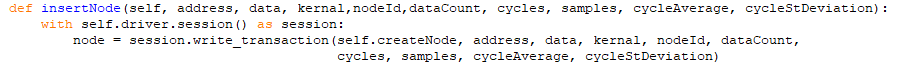
The first methods initialize the connection to the neo4j server and close the connections when they are called:



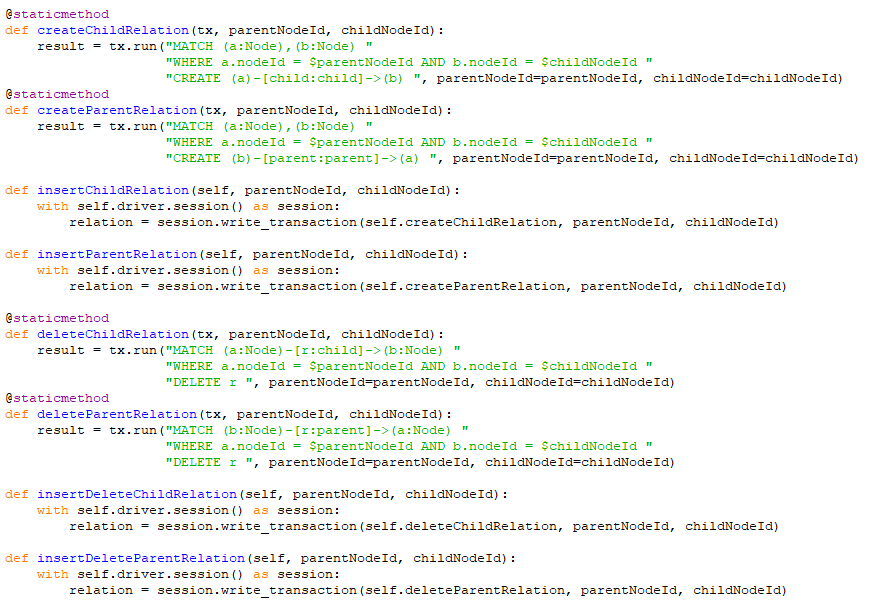
The node insertion is separated into 2 functions, one that contains the query that is in the Cypher Query Language that neo4j uses and another one that takes the data and uses the query code to write to the database.

Contains query: 

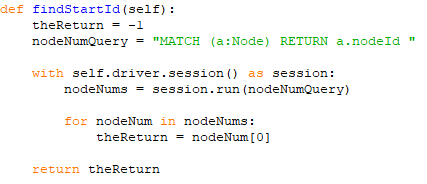
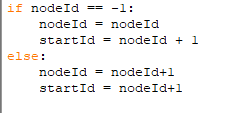
Writing is done using the query and the information is passed into this function:



Queries used for creating and deleting parent and child relationships are important and are done with queries. The writing function is called, and the values passed are used in the query:



The findStartId function is very important for inserting multiple trees. The start id is found in the database and is made the nodeId. To sperate different trees, the nodeId on the roots are 2 increments above the last nodeId:



All functions that are commented with NOT USED, work but have no use in the program.

The class named Node contains all of the information for a node that is inserted into the tree. This information includes the data, address, and kernel of the process. It also contains the cycle count, cycle average, and cycle standard deviation.

The class named Tree contains all functions related to insertion into the tree and any tree manipulation:

The createNode function is called to create a node. Then, the insertChild function is called to insert that create node into the tree. No duplicate nodes are allowed in the same path, so a check is done and if the node exists, the count of the node will be updated.

The insertIntoGraphDFS function is what takes the tree and inserts it into the graph database. The function calls insertNode for each tree node and relations for parent and child are created in the graph database. The function is recursively called until all nodes in the tree are inserted into the graph database.

The insertCycles function updates the cycle count, cycle average, and cycle standard deviation of the leaf node.

No other functions are used in the class

In the main function, the root is created for the table, the file for the process is read into the program in reverse. The line read in is split into the parts for its address, data, and kernel. These values are then stored in variables and inserted into the tree.

For the end of a sample, there is code dealing with the cycles. There is a check for if there are any processes or not, also if it was only one process compared to more than one. If no processes, then cycle insertion is skipped. If one process exists, the cycle is inserted and the tree is set back to its root., however a placeholder is also inserted into the tree before it is set back to its root due to the mechanics of insertion into the graph. This placeholder will be rectified at insertion into the graph. If there is more than one process, the cycles are inserted, and the tree is set back to its root. The setting back to the root allows for a new sample to be inserted.

In the program, the connection to the graph database is established with:



The connection to the graph database is closed with:



The Print or Delete Neo4j.py program either deletes all nodes and relationships or prints all nodes from the graph database. Uncomment the neo.printNodes to print, it is currently set to delete all nodes and relationships.



The Query.py program a starting nodeId to the graph database and returns the path of that node to the root. It finds the next nodeId above it and using recursion passes that nodeId to the same function. That will find the next nodeId and so on, until the root is reached.