GenomicGraphCore

Fred Criscuolo

3/1/23

Contents

Introduction

GenomicGraphCore is a set of backend applications that use Kotlin 1.7 components and Cypher Query Language (CQL) scripts to load core genomic data extracted from files downloaded from primary genomic data resources (*eg* UniProt, NCBI, GeneOntology) into a local Neo4j 5.n database to create a consistent core for adding more specialized entities (*eg* COSMIC).

> **!** Important
>
> In version 5, Neo4j revised the syntax for a number of Cypher commands. To run the CQL scripts in older Neo4j versions, some editing will be required.

It is anticipated that specialized graph schemas will link to the core schema via one or more relationships to the core's identifier node types (*i.e.* UniProtEntry, EntrezGene, and/or HGNC). The GenomicGraphCore database schema is intended to be readily extended by loading new data into new node and creating relationships to existing node types. For example, a specialized node type such as CosmicGene links the CosmicGraphDb components to the GenomicGraphCore components via its relationship to the latter's UniProtEntry nodes. There are many examples of integrating data from multiple sources into a graph database to model biological interactions. The design for the GenomicGraphCore database is based on work published by Lysenko *et al*.

Execution

The Genomic Core Neo4j graph database is created by loading ontology and genomic data from files downloaded from primary data sources such as UniProt and NCBI. The genomic data files are either tab or comma delimited and the data are loaded using CQL scripts that utilize the Neo4j LOAD CSV utility. By default, the CQL files are assumed to be in the application's data subdirectory. If that is not the case, the individual CQL scripts must be edited to reflect the correct file location. The ontology data are loaded from OBO-formateed files using a Kotlin applications. Currently, the GeneOntology and Human Phenotype Ontology data are downloaded by the application from their primary source into /tmpThe Squence Ontology does not support FTP access and its *so.obo* file is assumend to be in the application's data/so subdirectory.

As noted below, CQL scripts invoke custom stored procedures defined in the GenomicCore Neo4j Plugin library. This library must be copied into the Neo4j database plugins subdirectory.

The GenomicCoreDataLoader (*org.batteryparkdev.genomicgraphcore.cypher.app.GenomicCoreDataLoader*) application will define the Neo4j database constraints, load the ontology data, and then load the genomic data in the proper order. This Kotlin file should be edited to support new constraints, ontologies, and/or genomic data.

Implementation

As noted in the Indroduction, the ETL functions for this database are implemented in Kotlin and CQL. CQL scripts are utilized for data obtained in tab or comma delimited files. The *LOAD CSV* Neo4j function proved to be the most performative technique for loading these files. It is also robust in handling data anomalies in the source files. Data provided in non-delimited formats (*eg* OBO) are loaded using Kotlin components. The format of the CQL scripts was derived from similar scripts published by Tuck.

Requirements

1. Environment Properties For Neo4j The application requires the user to define a set of system environment properties.

env variable name	description	example
NEO4J_ACCOUNT	a registered Neo4j user	neo4j
NEO4J_PASSWORD	the password for the account	neo4j
NEO4J_URI	the Neo4j server connection URI	neo4j://localhost:7687
NEO4J_DATABASE	the specific NEO4J database	neo4j

> ℹ Note
>
> The NE04J_URI environment property can specify a remote Neo4j database server.

2. GenomicGraphCore Neo4j Plugin Library

Some of the application's CQl scripts invoke custom functions implemented in the GenomicCoreNeo4j github repository. After downloading or cloning this repository, users should build the library (* mvn clean package*) and copy the generated jar file to their Neo4j server's plugin directory. The Neo4j server should then be restarted to put the custom functions into scope

3. wisecubeai/pubmed-parser Library

Parsing of XML data loaded via RESTful requests to NCBI, as described below, requires the wisecube/pubmed-parser library be downloaded or cloned from github. After building and packaging this library, the jar file should be placed within the application's claspath.

PubMed data

Many of the node types within the core database have PubMed identifiers as properties. These properties are intended to provide reference data for the node. NCBI supports a RESTful API that will download PubMed data in a specified format (*eg* XML, JSON) for either individual PubMed IDs or batches of PubMed IDs. NCBI enforces a rate request frequency; 3 requests/second for non-registered users and 10 requests/second for registered users. The returned XML data is parsed by classes in the open-source wisecube/pubmed-parser library into JVM objects. Kotlin components map these JVM components into Cyher commands to create Neo4j nodes and relationships. Because accessing PubMed data via RESTful requests is significantly slower than loading data from delimited files, and a need to accomodate periodic NCBI service outages, PubMed nodes are loaded independently from other data sources. When a PubMed ID is encountered in loading a data file, a placeholder PubMed node, with the PubMed ID as an identifier, is created in the Neo4j database and a relationship between the source node and the PubMed node is created. For example UniProtEntry nodes may have several PubMed IDs as references. Placeholder PubMed nodes are created for any of the PubMed IDs that are novel. In addtion, a HAS_PUBLICATION relationship is created between the new UniProtEntry node and the PubMed node rgardless of whether it is novel or pre-existing. This methodlogy does not slow down or disrupt loading non-PubMed data. Two (2) standalone Kotlin applications are available to fetch additional data for these placeholder PubMed nodes (*eg* title, authors, abstract, *etc*). The first, *PubmedPropertiesLoader*, scans the Neo4j database for placeholder PubMed nodes and sends batch requests for PubMed data to NCBI. The returned PubMed data is used to complete the PubMed nodes and to establish a second level of placeholder Reference nodes for the references cited in a PubMed node. A second application, *PubmedReferenceLoader* performs the same data loading process for incomplete Reference nodes, but does not create another layer of Reference nodes. An NCBI outage will terminate loading PubMed data, but loading can be resumed without any manual intervention once service is restored. Placeholder nodes are processed regardless of their source and either or both of these loading applications can be run concurrently with other data loading operations. In Cypher, this structure can be represented as: **(u:UniProtEntry)-[r1:HAS_PUBLICATION]->(p:PubMed)-[r2:HAS_REFERENCE]->(ref:Reference)** Once invoked, these auxillary applications will run continuously until they can no longer find placeholder nodes in the database or access to the NCBI RESTful service is disrupted.

Data Resources

1. UniProt
2. NCBI Gene
3. GeneOntology
4. Human Phenotype Ontology
5. DisGeNET
6. HGNC
7. Reactome
8. DAVID Bioinformatics Resources
9. PubMed
10. David Tuck Dataverse

Acknowledgements

The contents of the GenomicGraphCore Neo4j database are loaded from the following services:

1. Sayers, Eric W et al. “Database resources of the National Center for Biotechnology Information in 2023.”

Nucleic acids research vol. 51,D1 (2023): D29-D38. doi:10.1093/nar/gkac1032

2. UniProt Consortium. “UniProt: the Universal Protein Knowledgebase in 2023.” Nucleic acids research vol. 51,D1 (2023): D523-D531. doi:10.1093/nar/gkac1052
3. Seal, Ruth L et al. “Genenames.org: the HGNC resources in 2023.” Nucleic acids research vol. 51,D1 (2023): D1003-D1009. doi:10.1093/nar/gkac888 4.Janet Piñero, Juan Manuel Ramírez-Anguita, Josep Saüch-Pitarch, Francesco Ronzano, Emilio Centeno, Ferran Sanz, Laura I Furlong, The DisGeNET knowledge platform for disease genomics: 2019 update, Nucleic Acids Research, Volume 48, Issue D1, 08 January 2020, Pages D845–D855, https://doi.org/10.1093/nar/gkz1021
4. [National Human Genome Research Institute]https://www.genome.gov/
5. Ashburner M, Ball CA, Blake JA, et al. Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. Nat Genet. 2000;25(1):25-29. doi:10.1038/75556
6. Gene Ontology Consortium. The Gene Ontology resource: enriching a GOld mine. Nucleic Acids Res. 2021;49(D1):D325-D334. doi:10.1093/nar/gkaa1113
7. Sebastian Köhler, Michael Gargano, Nicolas Matentzoglu, Leigh C Carmody, David Lewis-Smith, Nicole A Vasilevsky, Daniel Danis, Ganna Balagura, Gareth Baynam, Amy M Brower, Tiffany J Callahan, Christopher G Chute, Johanna L Est, Peter D Galer, Shiva Ganesan, Matthias Griese, Matthias Haimel, Julia Pazmandi, Marc Hanauer, Nomi L Harris, Michael J Hartnett, Maximilian Hastreiter, Fabian Hauck, Yongqun He, Tim Jeske, Hugh Kearney, Gerhard Kindle, Christoph Klein, Katrin Knoflach, Roland Krause, David Lagorce, Julie A McMurry, Jillian A Miller, Monica C Munoz-Torres, Rebecca L Peters, Christina K Rapp, Ana M Rath, Shahmir A Rind, Avi Z Rosenberg, Michael M Segal, Markus G Seidel, Damian Smedley, Tomer Talmy, Yarlalu Thomas, Samuel A Wiafe, Julie Xian, Zafer Yüksel, Ingo Helbig, Christopher J Mungall, Melissa A Haendel, Peter N Robinson, The Human Phenotype Ontology in 2021, Nucleic Acids Research, Volume 49, Issue D1, 8 January 2021, Pages D1207–D1217,

https://doi.org/10.1093/nar/gkaa1043

8. Tuck, D. A cancer graph: a lung cancer property graph database in Neo4j. BMC Res Notes 15, 45 (2022). https://doi.org/10.1186/s13104-022-05912-9
9. B.T. Sherman, M. Hao, J. Qiu, X. Jiao, M.W. Baseler, H.C. Lane, T. Imamichi and W. Chang. DAVID: a web server for functional enrichment analysis and functional annotation of gene lists (2021 update). Nucleic Acids Research. 23 March 2022. 50(W1):W216-W221. doi:10.1093/nar/gkac194. 11.Lysenko A, Roznovăţ IA, Saqi M, Mazein A, Rawlings CJ, Auffray C. Representing and querying disease networks using graph databases. BioData Min. 2016 Jul 25;9:23. doi: 10.1186/s13040-016-0102-8. PMID: 27462371; PMCID: PMC4960687.