Create BioCyc GDS databases

June 23, 2022

1 Create BioCyc GDS databases

github project: kg-prototypes
branch: graphdb_load

Steps

1. Download individual organism data files.
   http://bioinformatics.ai.sri.com/ecocyc/dist/flatfiles-52983746/

Curently downloaded files are listed in src/config/config.yml file - EcoCyc: ecoli_25.5.tar.gz - HumanCyc: human.tar_25.5.gz - YeastCyc: yeastcyc_25.5.tar.gz - PseudomonasCyc: pput160488cyc_25.5.tar.gz - BsubCyc: bsub_47.tar.gz

2. Parse the .dat files and process and write formatted data into .tsv files and put into one zip file for liquibase to load into Neo4j graph database
3. Put all post-load scripps into src/config/cypher/biocyc-cypher.yml file, except organism specific cypher queries
4. Generate liquibase changelog files
5. Move changelog files to migration/liquibase/{dbname}/changelogs folder, and rename.
6. Run liquibase update

Pre-requisition: Since GDS need enzyme name information. Enzyme database will be loaded into neo4j before loading biocyc data. The enzyme data will be removed one the enzyme name information is copied to reaction property (displayName)

[1]:
```
import os
import sys
root = os.getcwd().split('notebook')[0]
sys.path.append(os.path.join(root, 'src'))

from biocyc.biocyc_parser import *
from biocyc import biocyc_liquibase
from biocyc import bsubcyc_liquibase
from common.constants import *
```

1.1 Parse biocyc data files and generated processed tsv files

data source Download the data file from biocyc or https://portal.azure.com/#view/Microsoft_Azure_FileStora
5882-4572-8560-af80d7df69b5%2FresourceGroups%2Flifelike-ecosystem%2Fproviders%2FMicrosoft.Storage%2Fsto
graph/protocol/SMB

data files need to put into the input dir: {kg-prototypes}/graph-db/extraction/data/download/biocyc

output dir Parser outputs are written in {kg-prototypes}/graph-db/extraction/data/processed/biocyc.
A zip file be generated in the format {biocyc_dbname}-data-{version}.zip under {output dir}/{biocyc_dbname}, where biocyc_dbname is EcoCyc, HumanCyc etc.

The biocyc_dbname/data-source-file mapping is in {kg-prototypes}/graph-db/extraction/src/config/config.yml

[2]:
```
# set the biocyc_db to process
biocyc_db = DB_ECOCYC

parser = BiocycParser(biocyc_db)
# parser.parse_and_write_data_files()
```

1.2 Generate Liquibase changelogs file

Liquibase changelog generator expects the input file to be the output zip file from previous step. 5 changelog files will be generated.

1. init-changelog: the scripts will read the parser output zip file, and load data into neo4j database
2. post load changelogs: this is part of Lifelike biocyc database updates
   - link genes to NCBI gene
   - set node display name
   - set node description, including reaction description as an equation without stoichiometry
   - set reaction enzyme_name
   - set pathways for gene (for annotation)
   - set node property entityType
3. gds no-collapse changelogs: This is used to generate non-collapse gds database
   - all changes listed in #2
   - correction reaction input and output directions
   - create reversed reactions for reversible reactions if input and output are not the same. Put postfix '_r' in the reversed reaction eid and displayName
   - reverse gene to TranscriptionUnit relationship as (TranscriptionUnit)-[:HAS_GENE]->(Gene)
   - delete DNA binding site nodes
   - delete TYPE_OF relationships
   - remove orphan BioCycClass nodes
   - remove Enzyme nodes (from Enzyme database)

- set node synonyms property and remove Synonym nodes
- label some compounds as CurrencyMetabolite
- change description property to detail since description is used by sankey for other purpose

4. gds reg-collapse chanagelogs: This is used to generate reg-collapse gds database. The database will have parallel edges, therefore need to use multi-graph to run the analysis and traces
   - all changes listed in #3
   - Collapse reglations
     - for reglation with mode '+', change the relationship to 'ACTIVATES' then remove regulation node
     - for reglation with mode '-', change the relationship to 'INHIBITS' then remove regulation node for reglation with mode '', change the relationship to 'REGULATES' then remove regulation node
5. gds changelogs: This is used to create the general gds database. It has Regulation and EnzReaction nodes collapsed and removed. Multi-graph is needed to run the analysis and traces.
   - all changes listed in #4
   - collapse EnzReaction
     - for EnzReaction regulations, move the regulator to regulate the reactions directly
     - for EnzReaction catalyzes, move the protein to catalyze reactions directly
     - delete all EnzReaction nodes

[5]:
```
zip_datafile = 'EcoCyc-data-25.5.zip'
biocyc_dbname = DB_ECOCYC
author = 'rcai'  # change to your name
biocyc_liquibase.generate_changelog_files(zip_datafile, biocyc_dbname, author)
```

[7]:
```
zip_datafile = 'BsubCyc-data-47.zip'
biocyc_dbname = DB_BSUBCYC
author = 'rcai'
bsubcyc_liquibase.generate_changelog_files(zip_datafile, biocyc_dbname, author)
```

1.3 Run liquibase update

1. Install liquibase Follow steps in {kg-prototype}/graph-db/migration/liquibase-src/README.md to install liquibase, jar files, including the custom java classes.

2. Set up database folder for changelogs

- copy changelog-mater.xml
- copy liquibase.properties, and change the database settings (url, password)
- add changelog files into the folder changelogs, and order the changelog files as changelog-xxxx. Liquibase updates will be based on the changelog file name sequence

1.3.1 3. Run liquibase update

- create an database in neo4j with name matching the liquibase.properties
- run command:

```
liquibase --log-level=info update
```

2 Add new Biocyc GDS databases (different organism)

1. Download the data file from biocyc, and put into {data_dir}/download/biocyc folder
2. Add a new db variable in common/constants.py
3. In config/congig.yml file, add the dbname-filename mapping
4. Run BiocycParser.parse_and_write_data_files
5. Run biocyc_liquibase.generate_changelog_files
6. create database in neo4j, and run liquibase update

[ ]: