Data Sources:

* Sample Cases:
  + Simplest Case:
    - Looking at one of the CTD files (eg. <http://ctdbase.org/reports/CTD_chemicals_diseases.csv.gz>), this is the simplest scenario we thought of. The idea here is we declare the type of “Driver” we’d like to use from our set (the CSV driver), and going through each row we will select columns based on specs provided. Some of them would be used as Curies (ids) for nodes. We will call the columns data elements. So the problem can be formulated as collecting these, elements applying transformations (if indicated to make other elements) and organizing this elements to Node properties and edge properties.
  + Across multiple files:
    - Case of FooDb:
      * FooDb is a bit dispersed, for finding food-> chemical relation’s we had to cross over multiple files CSV files and relate them using certain columns as foreign keys. Since these files don’t have explicit definition of what foreign key is we need way of representing it. Once we do this, and covert the multi-file data into a flat format, the problem becomes as simple as CTD’s case.
    - APIs:
      * This approach where we have to jump around endpoints and grabbing some part for the next to chain together outputs to a single row.
      * This I think has the potential to mix up different types of data sources together, one element might be coming from an API the other coming from a CSV and so forth. KEGG is a good example where we parameterize the outputs of reactions to resolve relationships.
  + SparQL and SQL :
    - Once the drivers for these are set up, they are capable of generating table sets that we can easily parse over in similar approach of our simplest case.

High level workflow…

