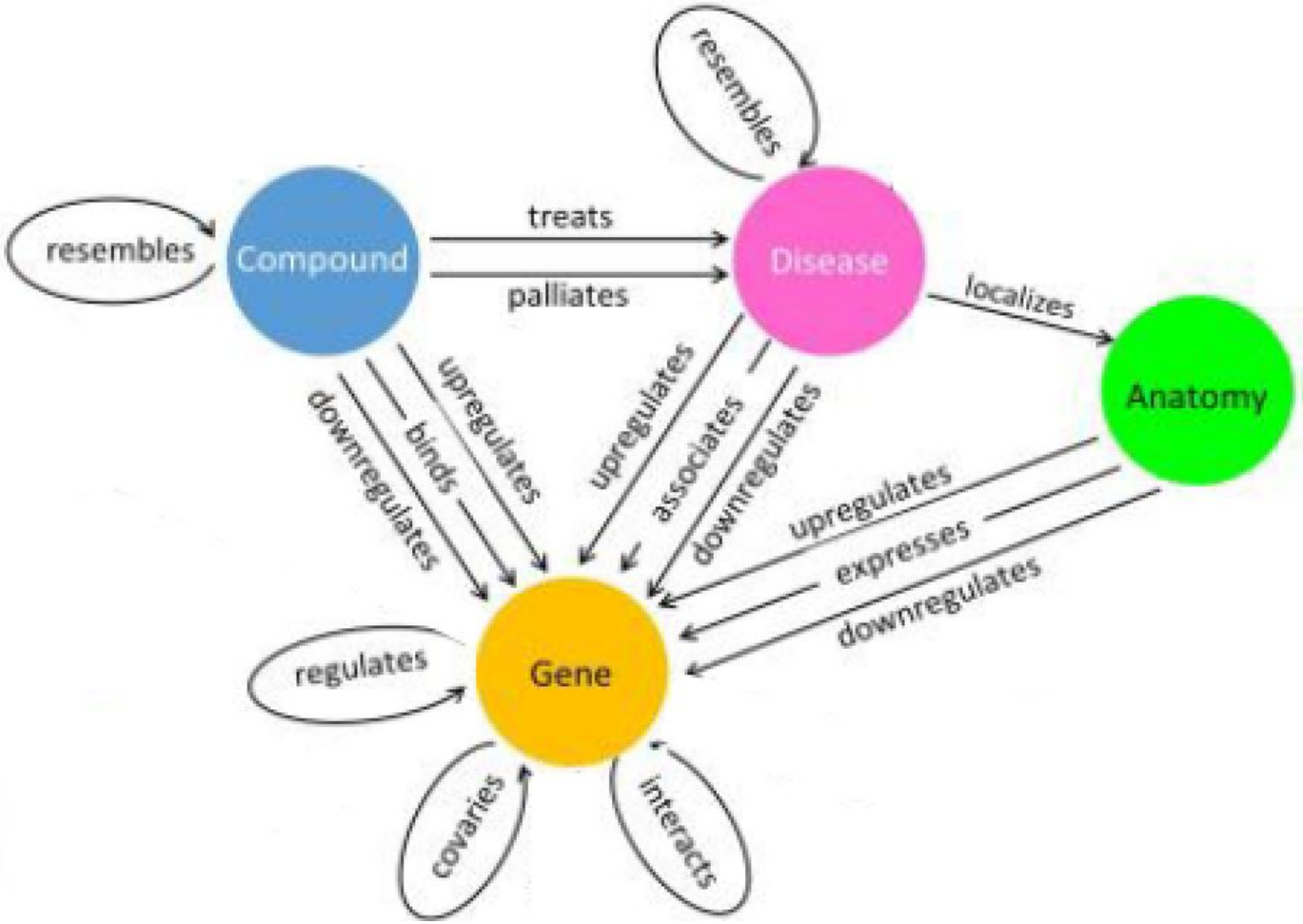
Project I: User Case

- HetioNet



Project I: User Case

- HetioNet
- Nodes

nodes.tsv

Id	name	kind
Gene::9997	SC02	Gene
Compound::DB09028	Cytisine	Compound

- Relationships

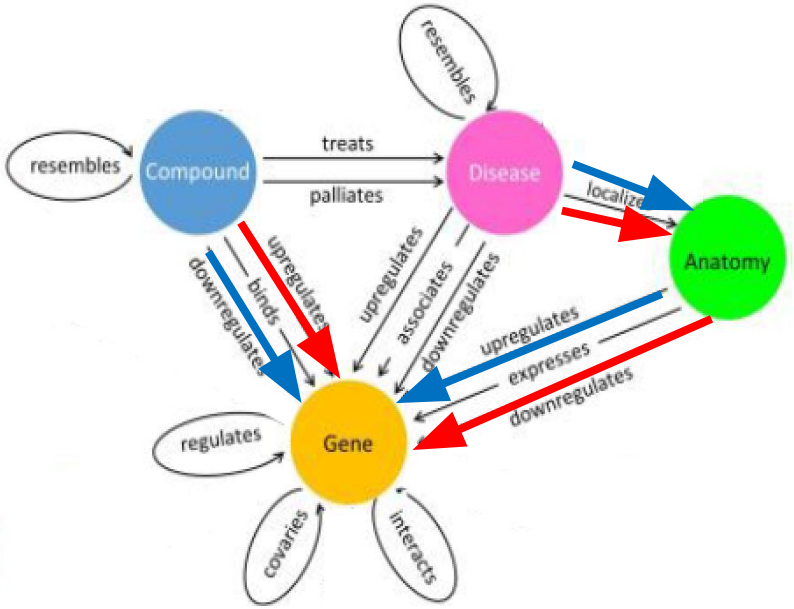
edges.tsv

source	metaedge	target
Gene::801	GiG	Gene::7428
Disease::DOID:263	DuG	Gene::857

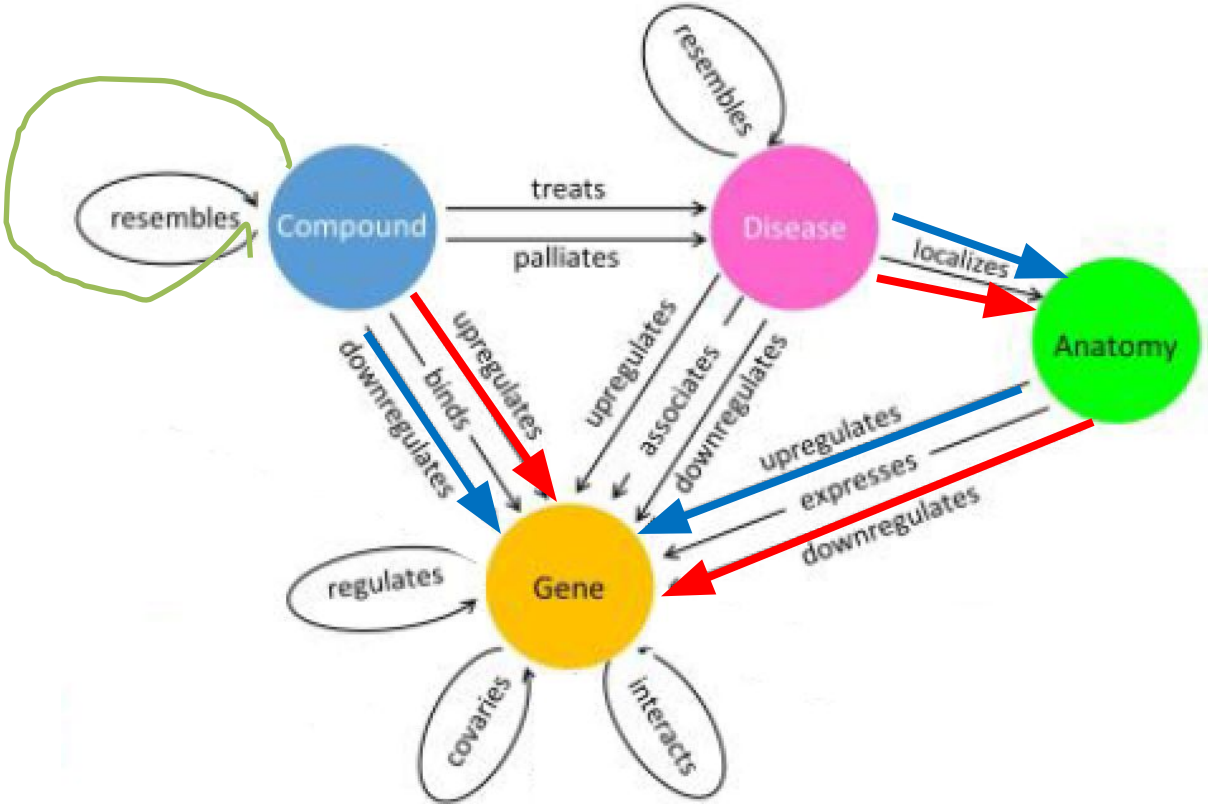
Project I: Requirement

- Build a database system to model *HetioNet*
- The database should at least answer the following questions in the quickest response time

  1. Given a disease id, what is its name, what are drug names that can treat or palliate this disease, what are gene names that cause this disease, and where this disease occurs? Obtain and output this information in a single query.



2. We assume that a compound can treat a disease if the compound up-regulates/down-regulates a gene, but the location down-regulates/up-regulates the gene in an opposite direction where the disease occurs. Find all compounds that can treat a new disease (i.e. the missing edges between compound and disease excluding existing drugs). Obtain and output all drugs in a single query.



Project I: Requirement

- A Python command-line client interface for database creation and query
- Use at least two types of NoSQL stores (Document, Graph, Key-value, Column Family)

Project I: Requirement

- Document (no hand-writing, in print!)
  - Design diagram
  - All queries
  - Potential improvements (e.g. how to speed up query)
- All source codes (upload to brightspace)
- Two-person team
- Due: 5 pm, March 17, Monday
- Project demo: 5:30 pm-8:15 pm, March 17 (Monday)

Project I: Rubric

- Database design: 30%
  - Rationale (15%)
  - Implementation (15%)
- Query functionality: 40%
  - 20% each query
- Client interface: 20% (GUI: 10 points)
- Presentation: 10%