Variant and Orthology Service

The goal of this document is to develop a schema design for the service in a graph database structure. A rough specification for the variant+ortholog service is also presented.

Rough Specifications

This page just lists some of our rough thoughts on what the service should do. Tim has also provided access to a set of scripts for converting the Ensembl data to a database: https://github.com/treynr/gripv

The key endpoint has the following inputs:

source species, source genome build, rs ID, target species, target genome build, filter on variant effects

Phase I

Return a list of genes associated with the variant (within, upstream cis, upstream trans, downstream cis, downstream trans)

There is an ISMB meeting. **submission for the tech track is due by May 9th, 2019.**

Genes

- make sure they are all the species that we want for GeneWeaver
- make a list of all the biotypes ordered by prevalence, Gaurab can help with definitions

Orthologs

- Getting the AGR orthology data will be important, but not necessary for phase I, other sources to follow, too

Transcripts

- make a list of all the biotypes ordered by prevalence, Gaurab can help with definitions

Variants

- keep all the alleles, not just the SNPs
- for now it's OK to just use the upstream/downstream assessment from Ensembl, this may change when we start adding epigenetic data

Queries

- query based on Ensembl IDs
- filter on variant_effect from the variant table
- user may have another identifier that we would have to have a mapping from that to the Ensembl ID. We will have to figure out how to handle all the mappings to/from GW IDs also.
- from human GWAS study, if we roll the variants up to the gene level, then how similar are the human and mouse genes?
    - How would we measure the similarity? Start with Jaccard similarity between the **ortholog cluster IDS** for the genesets.
- what are the variants between mouse and human for this given gene?

UI

- how to we want to show these data in GW?
- enumerate variants in the orthologous set?
- Extract from Elissa and Erich what they have in mind for the user interface.

Phase II

For each gene, also return a list of orthologs

Update the process for rapidly updating the builds. This should be done twice a year at a minimum.

Phase III

Provide a link / URL to mouse variant allele registry database (MVARD)

Phase IV

Provide a UI on top of the swagger-documented page.

Phase V

Include gene regulation information

This is other ways that variants impact genes. Tim is currently pulling data from the Ensembl regulatory build.

Additional Notes

We will need to update the GW database every 6 months.

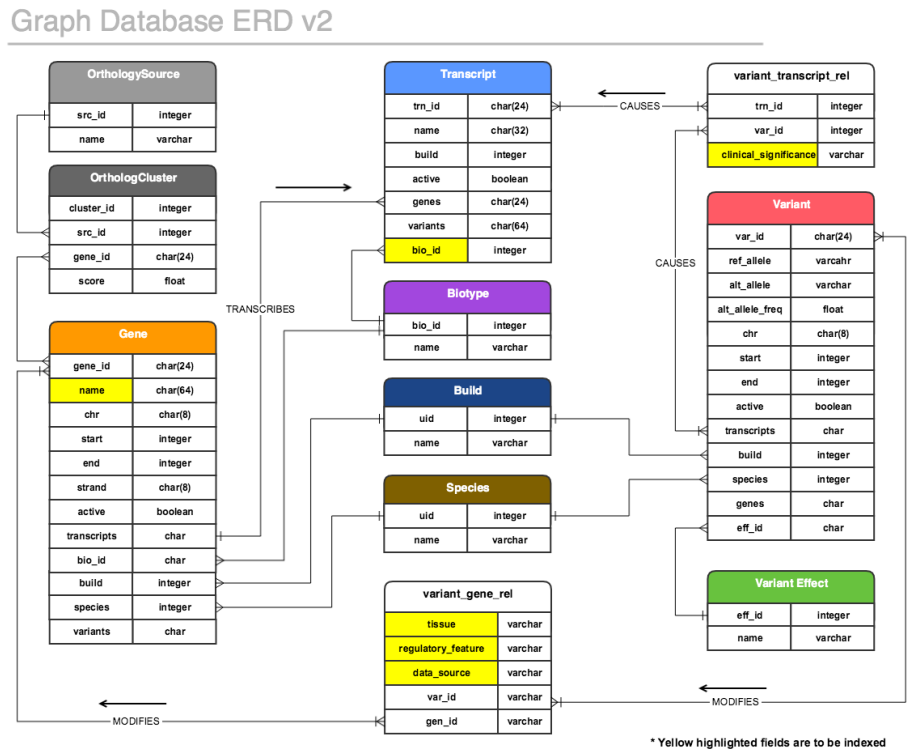
Each time we do a new build we will not throw anything away. Each new build will map the new build to existing genesets. We have to map everything, partial updates are no good.

Check out mappers from a relational database to a graph database.

Explore the use of neo4j (or other graph database) for storing the gene/ortholog/variant triples. Look at caps (upper bound) on the number of triples that we can store. Can postgres 11 handle this? We're dubious...

Data Model and Index Choices

The entity-relation diagram for the varolog graph database is shown below. Early in the development of the database we need to make some choices about which fields to index, and how to further normalize the graph DB.



Make sure we also store variants that may not be associated with a gene.

Use bio type to condition on the entities

Generic bipartite representation may not be the best approach for the long term (for all tools).

We can map biotypes on to one another.

Can this capture all the relevant regulatory relationships? TADs? Epigenetic marks? Chromatin structures? Distal enhancers? (Jake talk with Tim about this)

Database Indexes

Indexes in graph databases are a lot like those in relational databases. They do take up storage space, and they also significantly speed up query performance. Since theses relations will hold so much data (~200GB) we prefer to limit the number of indexes defined in the schema. Only the key query fields will be indexed. The tables below show the current values for the fields that we expect to index.

Indexed Fields

Variant_transcript_rel.clinical_significance

transcript	1338078
primary_transcript	1313212
ncRNA	517205
mRNA	285680
miRNA	40

Variants.variant_effect

intron_variant	253555
non_coding_transcript_variant	119122
downstream_gene_variant	109435
upstream_gene_variant	103619
NMD_transcript_variant	15548
non_coding_transcript_exon_variant	10755
3_prime_UTR_variant	8182
missense_variant	7338
synonymous_variant	5406
5_prime_UTR_variant	1874
frameshift_variant	282
coding_sequence_variant	248
stop_gained	178
inframe_deletion	141
splice_donor_variant	124
splice_acceptor_variant	103
inframe_insertion	57
start_lost	16
stop_lost	13
stop_retained_variant	5

mature_miRNA_variant	4
splice_region_variant	4
start_retained_variant	2
protein_altering_variant	1

Genes.biotype

protein_coding	41781
processed_pseudogene	19228
lincRNA	12800
antisense_RNA	8244
unprocessed_pseudogene	5205
TEC	4154
miRNA	4081
snRNA	3283
misc_RNA	2775
snoRNA	2450
processed_transcript	1320
sense_intronic	1214
transcribed_unprocessed_pseudogene	1056
rRNA	897
transcribed_processed_pseudogene	725
IG_V_gene	362
IG_V_pseudogene	343
TR_V_gene	252
sense_overlapping	217
bidirectional_promoter_lncRNA	153
TR_J_gene	149
polymorphic_pseudogene	141
transcribed_unitary_pseudogene	123
unitary_pseudogene	117
scaRNA	100
pseudogene	86

TR_V_pseudogene	64
IG_D_gene	56
Mt_tRNA	44
3prime_overlapping_ncRNA	33
IG_J_gene	32
ribozyme	30
IG_C_gene	27
TR_J_pseudogene	14
TR_C_gene	14
IG_C_pseudogene	10
TR_D_gene	8
sRNA	7
IG_LV_gene	4
Mt_rRNA	4
IG_D_pseudogene	3
IG_J_pseudogene	3
IG_pseudogene	3
non_coding	3
scRNA	2
translated_processed_pseudogene	2
macro_lncRNA	2
vaultRNA	1

Transcripts.biotypes

protein_coding	137554
retained_intron	47410
processed_transcript	43390
lincRNA	21425
nonsense_mediated_decay	20693
processed_pseudogene	19232
antisense_RNA	15144
unprocessed_pseudogene	5215
TEC	4330
miRNA	4081
snRNA	3283
misc_RNA	2792
snoRNA	2462

sense_intronic	1301
transcribed_unprocessed_pseudogene	1075
rRNA	897
transcribed_processed_pseudogene	732
IG_V_gene	445
sense_overlapping	392
IG_V_pseudogene	343
TR_V_gene	302
bidirectional_promoter_lncRNA	275
polymorphic_pseudogene	182
TR_J_gene	149
pseudogene	132
transcribed_unitary_pseudogene	125
unitary_pseudogene	117
non_stop_decay	111
scaRNA	100
TR_V_pseudogene	64
IG_D_gene	56
IG_C_gene	44
Mt_tRNA	44
3prime_overlapping_ncRNA	38
IG_J_gene	32
ribozyme	30
TR_C_gene	16
TR_J_pseudogene	14
translated_processed_pseudogene	14
IG_C_pseudogene	10
TR_D_gene	8
sRNA	7
IG_LV_gene	4
Mt_rRNA	4
non_coding	3
macro_lncRNA	3
IG_pseudogene	3
IG_J_pseudogene	3
IG_D_pseudogene	3
scRNA	2
vaultRNA	1

Testing Database

There is a graph database set up for development with the following connection parameters:

```
"http://gwdev01:7474/", username="neo4j", password="j4cks0nl4b"
```