# BIT05 – Database technologies

Jasper Anckaert

# Lecture 5 – Databases, APIs & version control

# Previously

- NoSQL
  - Non relational
  - More flexible
  - Column, key-value, graph, multi-model, document
- Document store
  - Semi-structured
  - Unique key
  - CRUD
  - No predefined data formats
  - No normalisation

# Previously

- MongoDB
  - Free and open-source
  - JSON-like documents
  - Sharding
  - GridFS

  - Database -> collection -> documents -> key-value pairs

  - Dot-notation
  - `_id` field

# Previously

- MongoDB – mongo shell
  - Create database
  - Create collection
  - Retrieve documents from collection
    - Query filter
      - Query operators
    - Query projection
      - Projection operators
    - Cursor modifier
  - Update operations
    - Update operators
  - Delete operations
  - Aggregation
    - Aggregation pipeline operators

# Previously

- MongoDB vs SQL

| MongoDB | SQL |
|---|---|
| database | database |
| collection | table |
| document | row |
| field | column |
| index | index |
| Primary key | Primary key |
| Embedded documents | joins |

- GUI – Robo 3T

# NoSQL

Exercises

- Import the protein coding genes from file protCodingGenes.bson

```
$ mongorestore --collection protCodingGenes --db test
protCodingGenes.bson
```

- Return the first 10 genes (alphabetically) on chromosome 22
- Return the last but one group of 10 genes (by position) on chromosome 12
- Return the number of unique gene names
- Return the 50 most common genes and their number of occurences
- Return a sorted list of the number of genes per chromosome

# Databases in bioinformatics

Why?

- Make biological data available to scientists
  - Collect data in a single place
  - Published data may be difficult to find (time-consuming)
- Make biological data available in computer-readable form
  - Needed for analysis

# Databases in bioinformatics

## Types of databases

- Characterisation based on several properties
  - Type of data
  - Data entry and quality control
  - Primary or derived data
  - Technical design
  - Maintainer status
  - Availability

# Databases in bioinformatics

## Types of databases

- Type of data
  - Nucleotide sequences
  - Protein sequences
  - Gene expression data
  - Metabolic pathways
  - 3D structures

# Databases in bioinformatics

## Types of databases

- Data entry and quality control
  - Data deposited directly
  - Appointed curators add and update data
  - Treatment of erroneous data: removed, or marked
  - Type and degree of error checking

# Databases in bioinformatics

## Types of databases

- Primary or derived data
  - Primary databases: experimental results
  - Secondary databases: results of analysis of primary databases
  - Aggregate of many databases
    - Consolidation of data
    - Combination of data

# Databases in bioinformatics

## Types of databases

- Technical design
  - Flat files
  - Relational database
  - Object oriented database

# Databases in bioinformatics

## Types of databases

- Maintainer status
  - Large, public institution (EMBL, NCBI)
  - Quasi-academic institute (Swiss Institute of Bioinformatics, TIGR)
  - Academic group or scientist
  - Commercial company

# Databases in bioinformatics

## Types of databases

- Availability
  - Publicly available, no restrictions
  - Available, but with copyright
  - Accessible, but not downloadable
  - Academic, but not freely available
  - Commercial

# Databases in bioinformatics

## Identifiers and accession codes

- Identify an entry in two different ways
  - Identifier
    - String of letters and digits (understandable)
    - Can usually change
  - Accession code (or number)
    - Number that uniquely identifies an entry in its database
    - Stable

# Databases in bioinformatics

Primary nucleotide sequence databases

- 3 main databases
  - EMBL (ENA), GenBank, DDBJ
  - Little error checking
  - Redundancy
  - Synchronized on a daily basis
  - No legal restrictions

# Databases in bioinformatics

Primary nucleotide sequence databases

- European Nucleotide Archive
  - DNA and RNA sequences
  - 3 databases
    - Sequence Read Archive
    - Trace Archive
    - EMBL Nucleotide Sequence Database
  - Maintained by European Bioinformatics Institue
  - XML, HTML, FASTA, FASTQ
  - http://www.ebi.ac.uk/ena

# Databases in bioinformatics

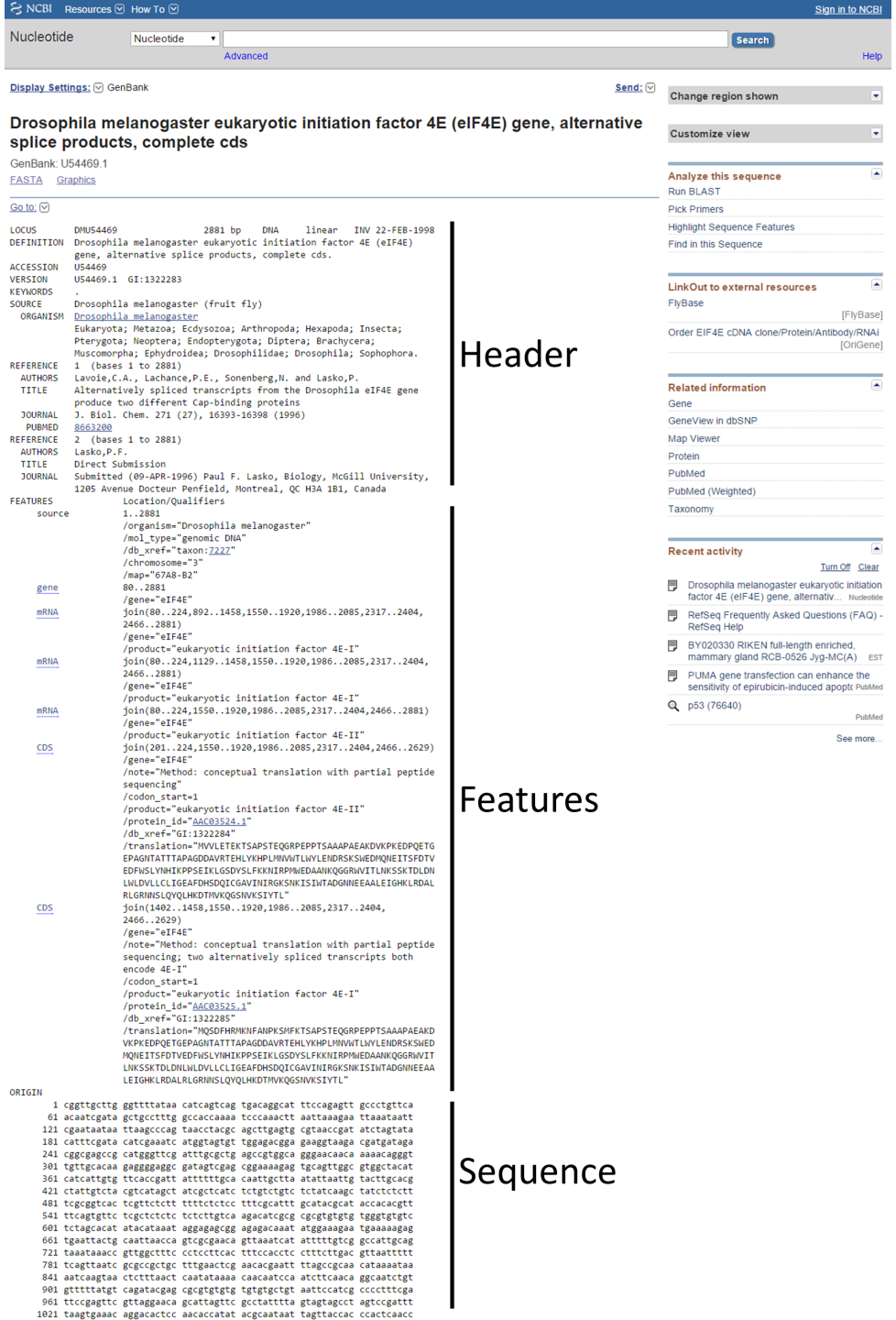
Primary nucleotide sequence databases

- GenBank
  - Open access
  - Publicly available nucleotide sequences and their protein translations
  - More than 100000 distinct organisms
  - Maintained by National Center for Biotechnology Information (NCBI)
  - Entries retrievable by NCBI GenBank webpage (or FTP)

# Databases in bioinformatics

## The GenBank format

- Flatfile with 3 main sections
  - Header
  - Features
  - Sequence



# Databases in bioinformatics

Primary nucleotide sequence databases


- DNA Data Bank of Japan
  - DNA sequences
  - Only nucleotide sequence data bank in Asia
  - http://www.ddbj.nig.ac.jp/

# Databases in bioinformatics

Secondary nucleotide sequence databases

- RefSeq
  - DNA, RNA and their protein products
  - Annotated and curated
  - Single record for each natural biological molecule
- OMIM
  - Catalog of human genes and genetic disorders and traits
  - Based on selection and review of published peer-reviewed literature
- HapMap
  - Haplotype map of the human genome
  - Genetic variants affecting health, disease and responses to drugs and environmental factors

# Databases in bioinformatics

Other nucleic acid databases

- Gene expression databases
  - Mostly microarray data
  - a.o. Gene Expression Omnibus, Expression Atlas, …
- Gene ontology
  - Relationships between concepts within a domain
- Genome databases
  - Annotated and analyzed genome sequences
  - a.o. Ensembl (Genomes), Flybase, Wormbase, …
- Phenotype databases
  - a.o. PhenCode
- RNA databases
  - a.o. miRBase, LNCipedia, …

# Databases in bioinformatics

## Sequencing databases

- Datasets from sequencing experiments
  - Sequence Read Archive
    - Hosted by NCBI
    - Raw data in BAM-format
    - Experimental metadata available
  - European Genome-phenome Archive
    - Hosted by EMBL-EBI
    - Data not publicly available

# Databases in bioinformatics

## Protein databases

- Protein sequence
  - Derived from translation of nucleotide sequences
    - secondary databases: NCBI Protein and trEMBL
  - Computational analysis, manual review and annotation
    - SwissProt
- Protein structure
  - a.o. Protein Data Bank, NCBI Structure

# Databases in bioinformatics

## The General Feature Format (GFF)

- Features of a particular gene, DNA and protein sequence
  - Tab-delimited
  - One line per feature, all but the final field in each feature line must contain a value; "empty" columns should be denoted with a '.'
    - **seqname** - name of the chromosome or scaffold; chromosome names can be given with or without the 'chr' prefix. **Important note**: the seqname must be one used within Ensembl, i.e. a standard chromosome name or an Ensembl identifier such as a scaffold ID, without any additional content such as species or assembly. See the example GFF output below.
    - **source** - name of the program that generated this feature, or the data source (database or project name)
    - **feature** - feature type name, e.g. Gene, Variation, Similarity
    - **start** - Start position of the feature, with sequence numbering starting at 1.
    - **end** - End position of the feature, with sequence numbering starting at 1.
    - **score** - A floating point value.
    - **strand** - defined as + (forward) or - (reverse).
    - **frame** - One of '0', '1' or '2'. '0' indicates that the first base of the feature is the first base of a codon, '1' that the second base is the first base of a codon, and so on..
    - **attribute** - A semicolon-separated list of tag-value pairs, providing additional information about each feature.

# Databases in bioinformatics

## The General Feature Format (GFF)

```
##gff-version 3
##sequence-region P69905 1 142
P69905  UniProtKB       Initiator methionine    1       1       .       .       .
Note=Removed;Ontology_term=ECO:0000269,ECO:0000269,ECO:0000269,ECO:0000269;evidence=ECO:0000269|PubMed:12665801,ECO:0000269|PubMed:13872627,ECO:0000269|PubMed:13954546,ECO:0000269|PubMed:14093912;Db
xref=PMID:12665801,PMID:13872627,PMID:13954546,PMID:14093912
P69905  UniProtKB       Chain   2       142     .       .       .       ID=PRO_0000052653;Note=Hemoglobin subunit alpha
P69905  UniProtKB       Metal binding   59      59      .       .       .       Note=Iron (heme distal ligand)
P69905  UniProtKB       Metal binding   88      88      .       .       .       Note=Iron (heme proximal ligand)
P69905  UniProtKB       Site    12      12      .       .       .       Note=Not glycated
P69905  UniProtKB       Site    57      57      .       .       .       Note=Not glycated
P69905  UniProtKB       Site    61      61      .       .       .       Note=Not glycated
P69905  UniProtKB       Site    91      91      .       .       .       Note=Not glycated
P69905  UniProtKB       Site    100     100     .       .       .       Note=Not glycated
P69905  UniProtKB       Modified residue        4       4       .       .       .       Note=Phosphoserine;Ontology_term=ECO:0000244;evidence=ECO:0000244|PubMed:24275569;Dbxref=PMID:24275569
P69905  UniProtKB       Modified residue        8       8       .       .       .       Note=N6-succinyllysine%3B alternate;Ontology_term=ECO:0000250;evidence=ECO:0000250|UniProtKB:P01942
P69905  UniProtKB       Modified residue        9       9       .       .       .       Note=Phosphothreonine;Ontology_term=ECO:0000244;evidence=ECO:0000244|PubMed:24275569;Dbxref=PMID:24275569
P69905  UniProtKB       Modified residue        12      12      .       .       .       Note=N6-succinyllysine;Ontology_term=ECO:0000250;evidence=ECO:0000250|UniProtKB:P01942
P69905  UniProtKB       Modified residue        17      17      .       .       .       Note=N6-acetyllysine%3B
alternate;Ontology_term=ECO:0000244;evidence=ECO:0000244|PubMed:19608861;Dbxref=PMID:19608861
P69905  UniProtKB       Modified residue        17      17      .       .       .       Note=N6-succinyllysine%3B alternate;Ontology_term=ECO:0000250;evidence=ECO:0000250|UniProtKB:P01942
P69905  UniProtKB       Modified residue        25      25      .       .       .       Note=Phosphotyrosine;Ontology_term=ECO:0000244;evidence=ECO:0000244|PubMed:24275569;Dbxref=PMID:24275569
P69905  UniProtKB       Modified residue        36      36      .       .       .       Note=Phosphoserine;Ontology_term=ECO:0000244;evidence=ECO:0000244|PubMed:24275569;Dbxref=PMID:24275569
P69905  UniProtKB       Modified residue        41      41      .       .       .       Note=N6-succinyllysine%3B alternate;Ontology_term=ECO:0000250;evidence=ECO:0000250|UniProtKB:P01942
P69905  UniProtKB       Modified residue        50      50      .       .       .       Note=Phosphoserine;Ontology_term=ECO:0000244;evidence=ECO:0000244|PubMed:24275569;Dbxref=PMID:24275569
P69905  UniProtKB       Modified residue        103     103     .       .       .       Note=Phosphoserine;Ontology_term=ECO:0000250;evidence=ECO:0000250|UniProtKB:P01942
P69905  UniProtKB       Modified residue        109     109     .       .       .       Note=Phosphothreonine;Ontology_term=ECO:0000250;evidence=ECO:0000250|UniProtKB:P01942
P69905  UniProtKB       Modified residue        125     125     .       .       .       Note=Phosphoserine;Ontology_term=ECO:0000250;evidence=ECO:0000250|UniProtKB:P01942
P69905  UniProtKB       Modified residue        132     132     .       .       .       Note=Phosphoserine;Ontology_term=ECO:0000250;evidence=ECO:0000250|UniProtKB:P01942
P69905  UniProtKB       Modified residue        135     135     .       .       .       Note=Phosphothreonine;Ontology_term=ECO:0000250;evidence=ECO:0000250|UniProtKB:P01942
P69905  UniProtKB       Modified residue        138     138     .       .       .       Note=Phosphothreonine;Ontology_term=ECO:0000250;evidence=ECO:0000250|UniProtKB:P01942
P69905  UniProtKB       Modified residue        139     139     .       .       .       Note=Phosphoserine;Ontology_term=ECO:0000250;evidence=ECO:0000250|UniProtKB:P01942
```

# Databases in bioinformatics

## Genome browsers

- Graphical interface for genomic data
  - UCSC genome browser
    - Search by gene name
    - Search by location (chrN:startposition-stoppostion)
  - Ensembl genome browser
    - Annotated genes aligned to a reference genome
    - Export data in multiple format (FASTA, GFF, EMBL, …)

# Databases in bioinformatics

Exercices

- What information about the rabies virus sequence can you obtain from its annotations in the NCBI Sequence Database? Give the accession number, definition, organism and PubMed ID of the record.
- How many nucleotide sequences are there from the bacterium Chlamydia trachomatis?
- How many nucleotide sequences are there from the bacterium Chlamydia trachomatis in the RefSeq part of the NCBI Sequence Database?
- How many nucleotide sequences were submitted to NCBI by Matthew Berriman?
- How many nucleotide sequences from the nematode worms are there in the RefSeq Database?

# Databases in bioinformatics

Exercices – part 2

- How many nucleotide sequences for collagen genes from nematode worms are there in the NCBI Database?
- How many mRNA sequences for collagen genes from nematode worms are there in the NCBI Database?
- How many protein sequences for collagen proteins from nematode worms are there in the NCBI database?
- What is the accession number for the Trypanosoma cruzi genome in NCBI?
- How many fully sequenced nematode worm species are represented in the NCBI Genome database?
- Find the accession number of human beta-globin mRNA sequence. What is the accession number of the encoded protein ? How many amino acids does it contain?

# Databases in bioinformatics

Exercices – part 3

- Find the publication in PubMed with the following ID: 8663200
  - Use the search record of this publication in PubMed to obtain this data entry in the Nucleotide database
  - Download the GenBank formatted flatfile
- Find the corresponding record from the previous exercise in the ENA database
  - Compare the EMBL format to the GenBank format you downloaded before
- How many alternative transcripts are know for Drosophila melanogaster eIF-4E
- Find the human hemoglobin alpha protein in UniprotKB. What is the entry name?
- How many genes are associated with Huntington Disease (HD), with Alzheimer's disease (AD) and with Parkinson's disease (PD)?
- How many transcripts does Ensembl predict for the human gene ACHE?
- Find the mouse orthologue of the human SSBP4. Does this gene have paralogues?

# Databases in bioinformatics

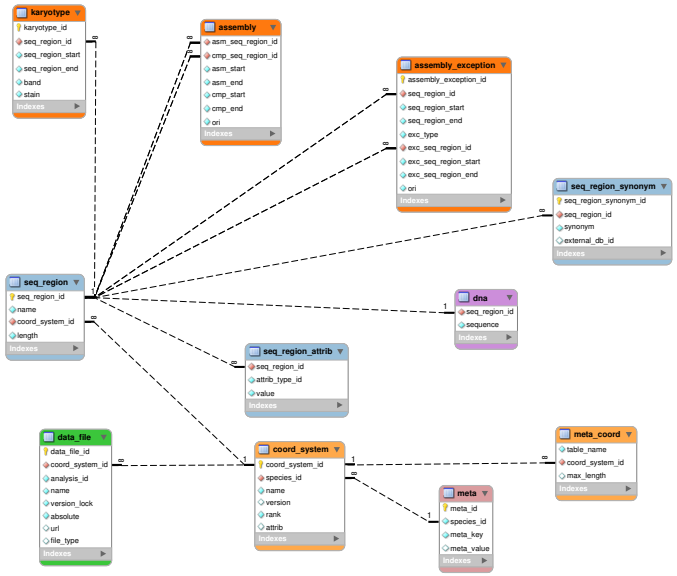
## Query MySQL databases directly

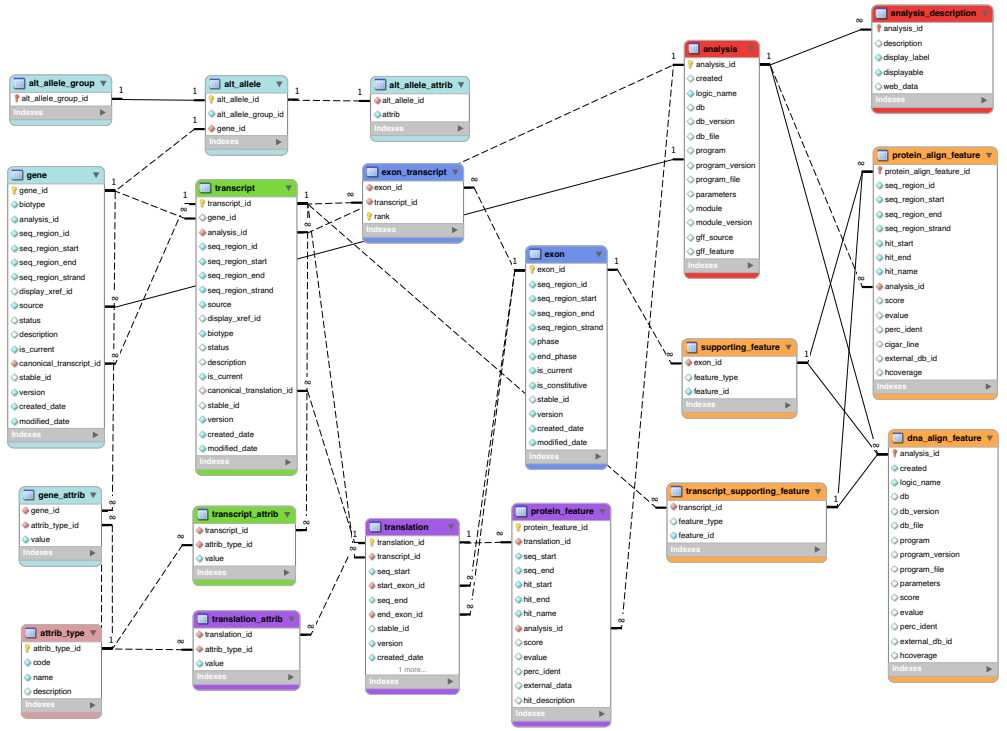
- UCSC
  - Hostname genome-mysql.cse.ucsc.edu
  - User genome
  - Password
- Gene Ontology
  - Hostname mysql-amigo.ebi.ac.uk
  - User go_select
  - Password amigo
  - Database go_latest
  - Port 4085
- Ensembl
  - Hostname ensembldb.ensembl.org
  - User anonymous
  - Password

# Databases in bioinformatics

## Ensembl database

- Complex database schemas
- Not suited to retrieve sequences



# Databases in bioinformatics

API

- Uniform method of access to data
- Reusable in different systems
- Reliable
- Insulates developers to underlying database changes

# Databases in bioinformatics

## Ensembl API

- Perl API
- Installation instructions on Ensembl website
- Different versions based on Ensembl release
- Use Registry to find Ensembl database and connect to them

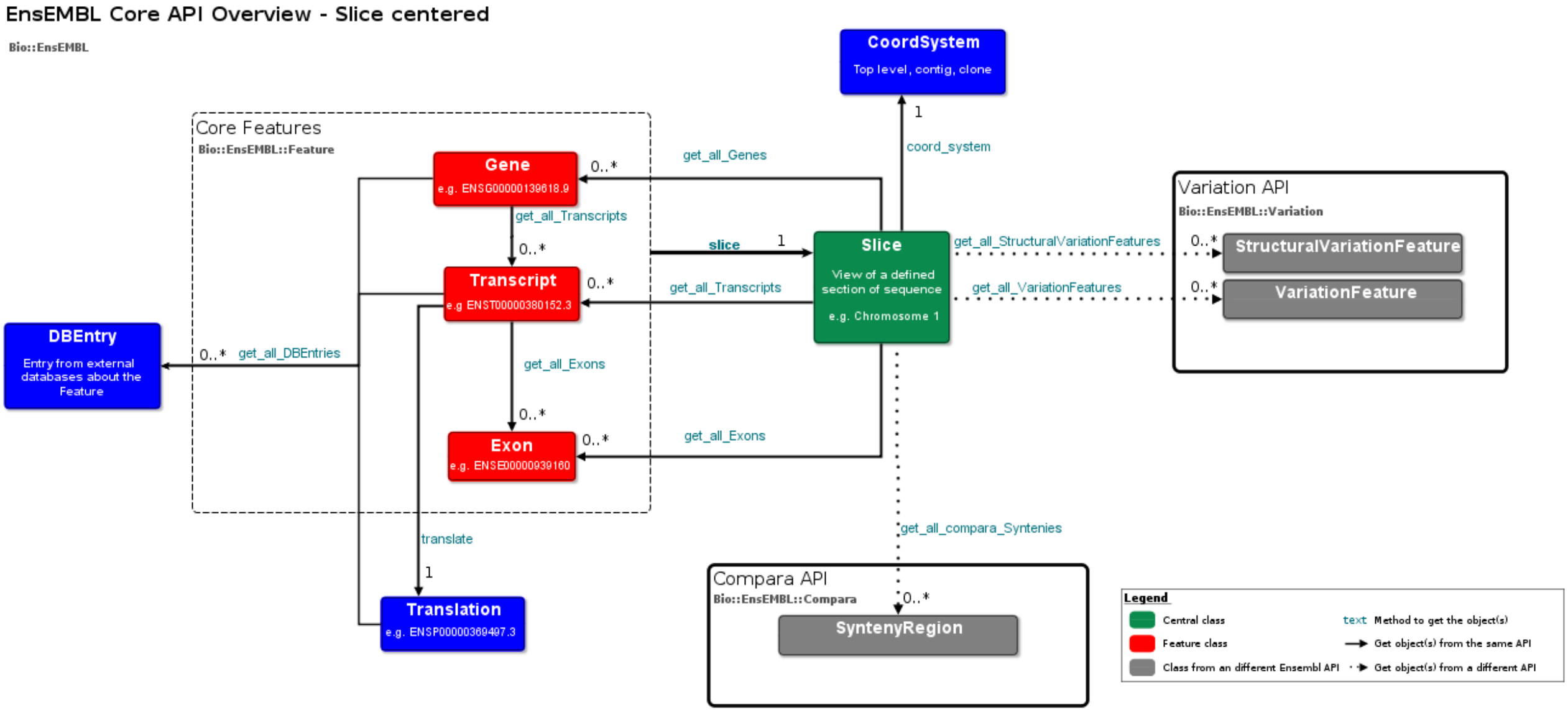
```
Bio::EnsEMBL::Registry->load_registry_from_db(
    -host => 'ensembldb.ensembl.org',
    -user => 'anonymous',
    -verbose => '1'
);
```

# Databases in bioinformatics

## Ensembl API

- Several databases
  - Core (genes, transcripts, translations, assembly, sequence)
  - Compara (SNVs, CNVs, somatic variations, phenotypes)
  - Variation (gene trees, homologies, multiple and pairwise genomic alignments)
  - Regulation (regulation, motifs, array probes)

# Databases in bioinformatics



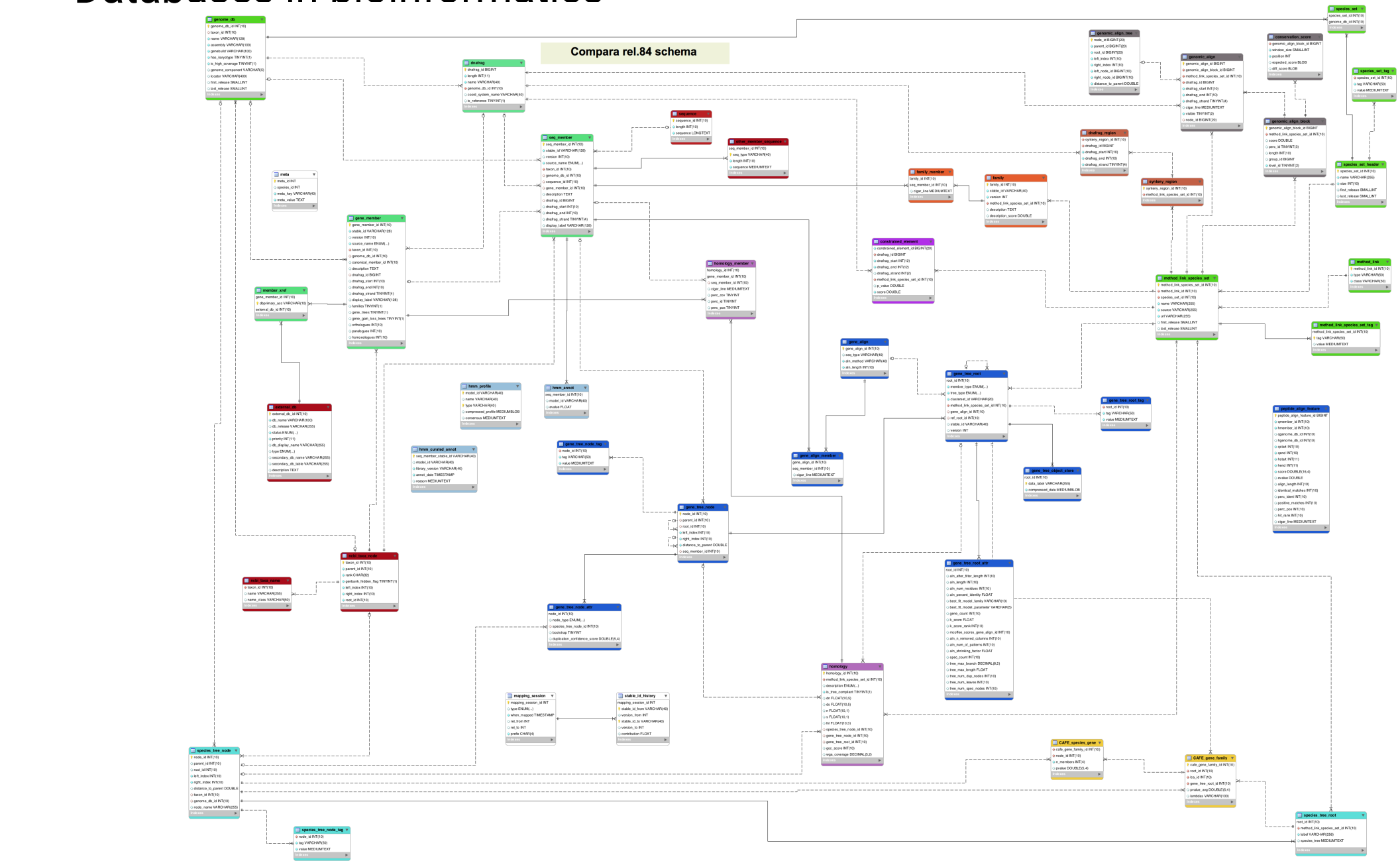
# Databases in bioinformatics

Ensembl API

- Core database
  - Annotation information for each organism in Ensembl
  - Species specific databases

```
# get a slice adaptor for the human core database
my $slice_adaptor = $registry->get_adaptor( 'Human', 'Core', 'Slice' );

# Fetch all clones from a slice adaptor (returns a list reference)
my $clones_ref = $slice_adaptor->fetch_all('clone');

# If you want a copy of the contents of the list referenced by
# the $clones_ref reference...
my @clones = @{$clones_ref};

# Get the first clone from the list via the reference:
my $first_clone = $clones_ref->[0];

# Iterate through all of the genes on a clone
foreach my $gene ( @{ $first_clone->get_all_Genes() } ) {
    print $gene->stable_id(), "\n";
}

# More memory efficient way of doing the same thing
my $genes = $first_clone->get_all_Genes();
while ( my $gene = shift @{$genes} ) {
    print $gene->stable_id(), "\n";
}

# Retrieve a single Slice object (not a list reference)
my $clone = $slice_adaptor->fetch_by_region( 'clone', 'AL031658.11' );
# No dereferencing needed:
print $clone->seq_region_name(), "\n";
```

# Databases in bioinformatics

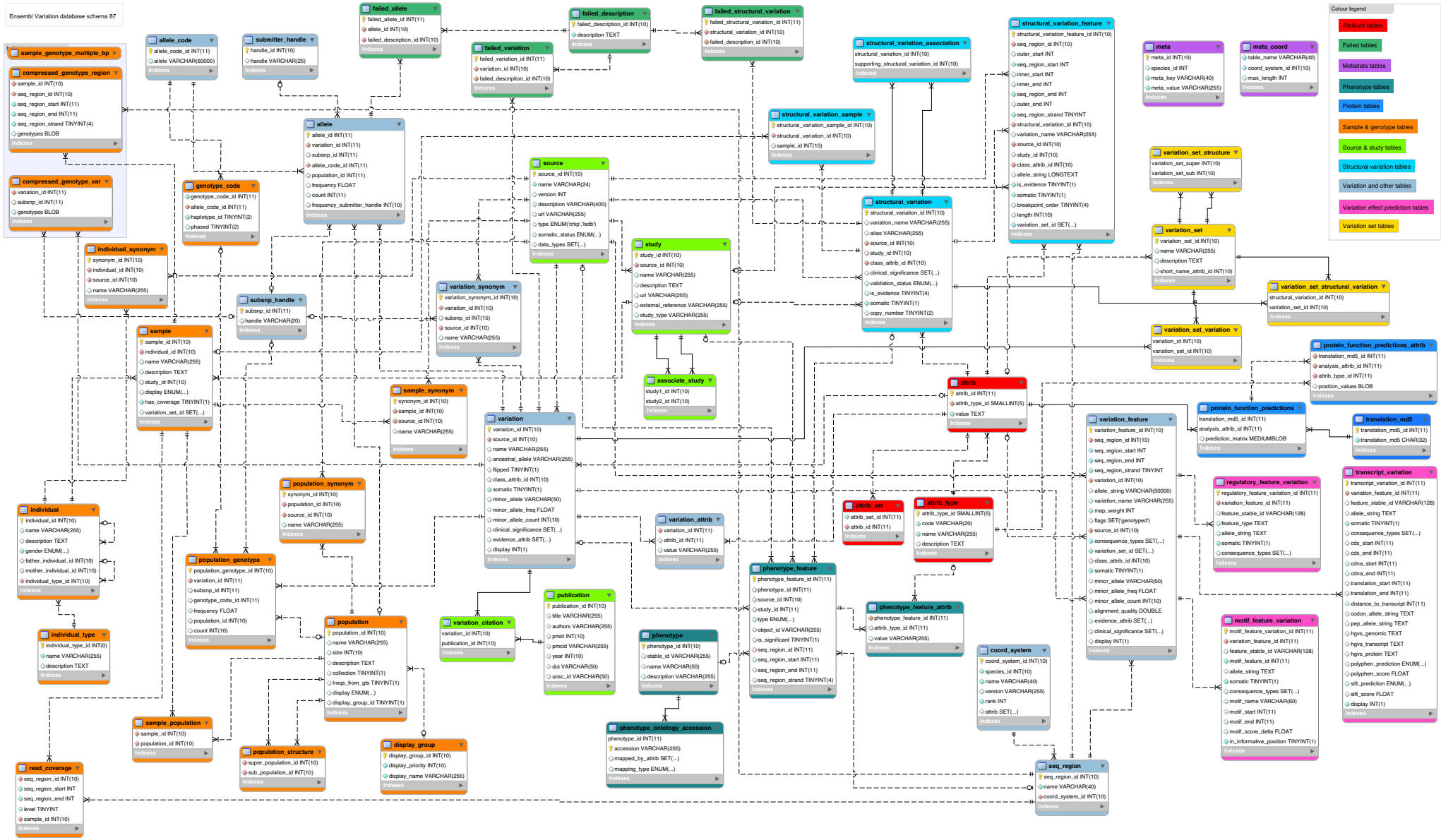


# Databases in bioinformatics

Ensembl API

- Compara database
  - Cross-species database
  - Genome-wide species comparisons
    - DNA-sequence level
      - Whole genome alignments
      - Synteny regions
      - Conservation scores / constrained elements
    - Gene level
      - Phylogenetic trees
      - Homology predictions

# Databases in bioinformatics

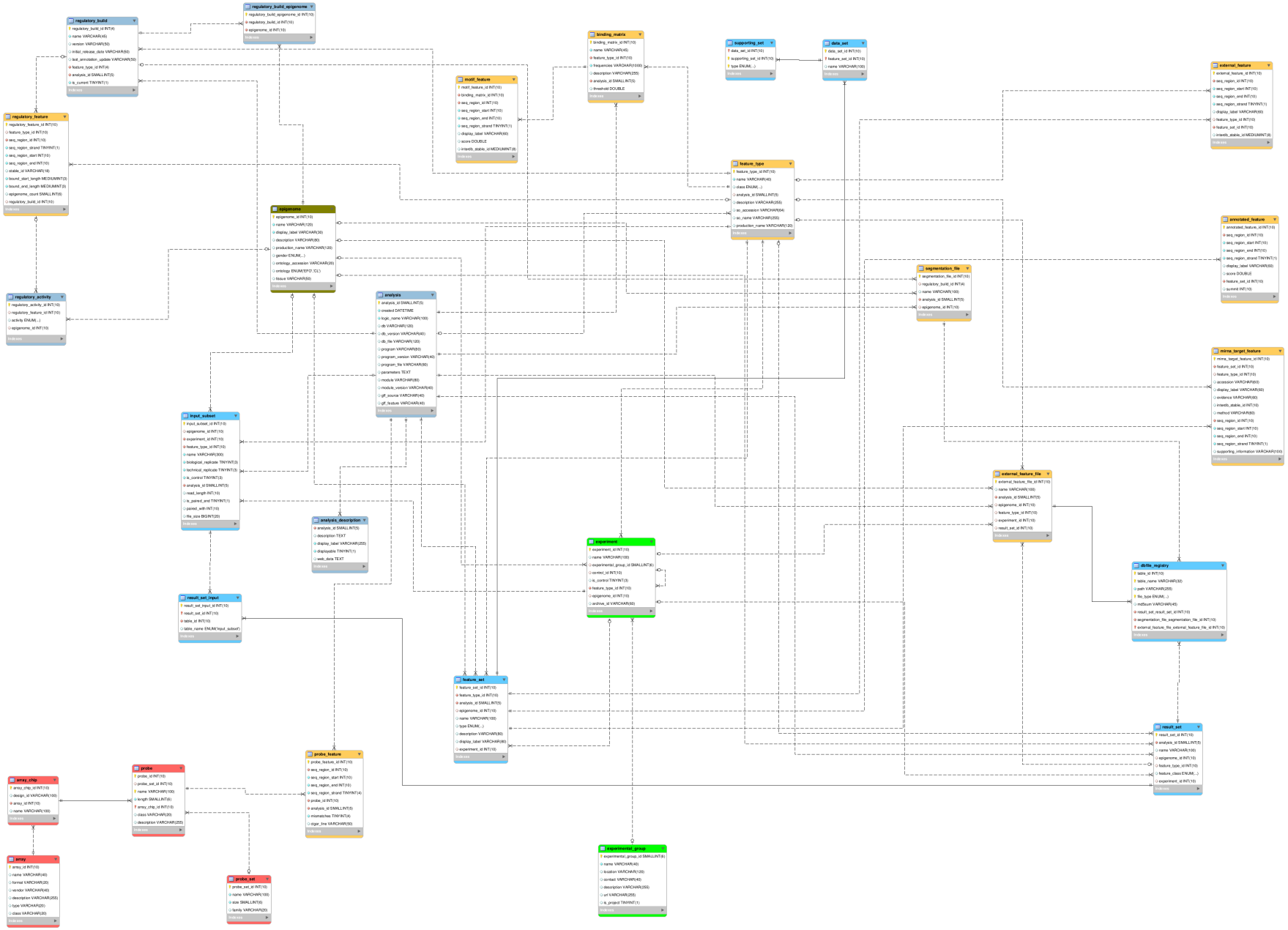


# Databases in bioinformatics

Ensembl API

- Variation database
  - Areas of the genome that differ between individual genomes
  - Associated disease and phenothpe information
  - Different types of variants
    - Sequence variants
      - SNP (Single Nucleotide Polymorphism)
      - Insertion (one or more nucleotides)
      - Deletion (one or more nucleotides)
      - Indel (insertion and deletion, affecting 2 or more nucleotides)
      - Substitution (no change in length)
    - Structural variants
      - CNV (Copy Number Variation)
      - Inversion
      - Translocation

# Databases in bioinformatics



# Databases in bioinformatics

Ensembl API

- Regulation database
  - Gene expression and its regulation in human and mouse
  - Focus on transcriptional and post-transcriptional mechanisms

# Databases in bioinformatics

REST API

- Representational state transfer / RESTful
  - Base URL
  - Internet media type
  - Standard HTTP methods
    - OPTIONS
    - GET: list or retrieve
    - PUT: replace or create
    - POST: create new entry
    - DELETE: remove

# Databases in bioinformatics

Ensembl REST API

- http://rest.ensembl.org
- Language agnostic bindings to Ensembl data

- Able to create REST client in
  - JAVA
  - Perl
  - Python
  - Ruby

# Databases in bioinformatics

## Ensembl REST API – Python

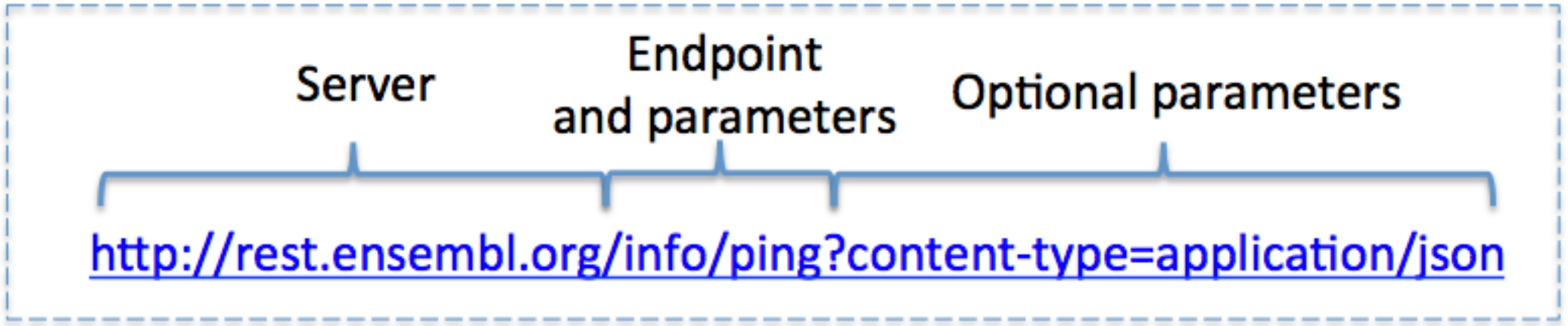
```python
#!/usr/bin/env python

import sys
import urllib
import urllib2
import json
import time

class EnsemblRestClient(object):
    def __init__(self, server='http://rest.ensembl.org', reqs_per_sec=15):
        self.server = server
        self.reqs_per_sec = reqs_per_sec
        self.req_count = 0
        self.last_req = 0

    def perform_rest_action(self, endpoint, hdrs=None, params=None):
        if hdrs is None:
            hdrs = {}

        if 'Content-Type' not in hdrs:
            hdrs['Content-Type'] = 'application/json'

        if params:
            endpoint += '?' + urllib.urlencode(params)

        data = None

        # check if we need to rate limit ourselves
        if self.req_count >= self.reqs_per_sec:
            delta = time.time() - self.last_req
            if delta < 1:
                time.sleep(1 - delta)
            self.last_req = time.time()
            self.req_count = 0

        try:
            request = urllib2.Request(self.server + endpoint, headers=hdrs)
            response = urllib2.urlopen(request)
            content = response.read()
            if content:
                data = json.loads(content)
            self.req_count += 1
```

```python
        except urllib2.HTTPError, e:
            # check if we are being rate limited by the server
            if e.code == 429:
                if 'Retry-After' in e.headers:
                    retry = e.headers['Retry-After']
                    time.sleep(float(retry))
                    self.perform_rest_action(endpoint, hdrs, params)
            else:
                sys.stderr.write('Request failed for {0}: Status code: {1.code} F

        return data

    def get_variants(self, species, symbol):
        genes = self.perform_rest_action(
            '/xrefs/symbol/{0}/{1}'.format(species, symbol),
            params={'object_type': 'gene'}
        )
        if genes:
            stable_id = genes[0]['id']
            variants = self.perform_rest_action(
                '/overlap/id/{0}'.format(stable_id),
                params={'feature': 'variation'}
            )
            return variants
        return None


def run(species, symbol):
    client = EnsemblRestClient()
    variants = client.get_variants(species, symbol)
    if variants:
        for v in variants:
            print '{seq_region_name}:{start}-{end}:{strand} ==> {id} ({consequenc

if __name__ == '__main__':
    if len(sys.argv) == 3:
        species, symbol = sys.argv[1:]
    else:
        species, symbol = 'human', 'BRAF'

    run(species, symbol)
```

# Databases in bioinformatics

## Ensembl REST API

- URL structure
  - 0 or more required parameters
  - 0 or more optional parameters
  - In a standard URL required parameters are flagged with a : e.g. :species.
  - Optional parameters should go into the request body if performing a POST or as key value pairs after the ? if performing a GET.



# Databases in bioinformatics

## Ensembl REST API

- curl

```
curl  ‘<url>’
      [-H ‘<header>’]
      [-X <request_method>]
      [-d ‘<data>’]
```

# Databases in bioinformatics

Ensembl REST API

- Parameters
  - Specify what is required and type of returned data from REST API

  - id
  - region
  - species
  - symbol
  - external_db
  - object_type
  - callback

# Databases in bioinformatics

Ensembl REST API

- Output formats
  - JSON, FASTA, BED, XML, …
  - Depends on client and operation
    - GET
      - `Content-type` HTTP header
      - `Content-type` HTTP parameter
      - `Accept` HTTP header
      - File extension
    - POST
      - `Accept` HTTP header

# Databases in bioinformatics

## Ensembl REST API – Endpoints

- Archive

| | |
|---|---|
| `GET archive/id/:id` | Uses the given identifier to return the archived sequence |
| `POST archive/id` | Retrieve the archived sequence for a set of identifiers |

- Comparative genomics

| | |
|---|---|
| `GET genetree/id/:id` | Retrieves a gene tree for a gene tree stable identifier |
| `GET genetree/member/id/:id` | Retrieves the gene tree that contains the gene / transcript / translation stable identifier |
| `GET genetree/member/symbol/:species/:symbol` | Retrieves the gene tree that contains the gene identified by a symbol |
| `GET alignment/region/:species/:region` | Retrieves genomic alignments as separate blocks based on a region and species |
| `GET homology/id/:id` | Retrieves homology information (orthologs) by Ensembl gene id |
| `GET homology/symbol/:species/:symbol` | Retrieves homology information (orthologs) by symbol |

# Databases in bioinformatics

## Ensembl REST API – Endpoints

- Variation

| | |
|---|---|
| `GET variation/:species/:id` | Uses a variant identifier (e.g. rsID) to return the variation features including optional genotype, phenotype and population data |
| `POST variation/:species` | Uses a list of variant identifiers (e.g. rsID) to return the variation features including optional genotype, phenotype and population data |

- Sequence

| | |
|---|---|
| `GET sequence/id/:id` | Request multiple types of sequence by stable identifier. Supports feature masking and expand options. |
| `POST sequence/id` | Request multiple types of sequence by a stable identifier list. |
| `GET sequence/region/:species/:region` | Returns the genomic sequence of the specified region of the given species. Supports feature masking and expand options. |
| `POST sequence/region/:species` | Request multiple types of sequence by a list of regions. |

# Databases in bioinformatics

Exercices

- Return the archived sequence with Ensembl id *ENSG00000141510*
- Return the archived sequence for both *ENSG00000012048* and *ENSG00000136997*
- Return a condensed XML-list of all orthologues in Mus musculus for *ENSG00000159763*
  - Do the same for BRCA2

  *HINTS:* *type=orthologues*
  *target_taxon=<taxon_id>*
  *format=condensed*
- Retrieve the genomic FASTA sequence for *ENST00000288602.10*
- Get a sequence from 100 nucleotides located on human chromosome 2 starting at position 100000
- Show the taxonomy information of the mouse
- Find the species and the database for *ENSMUSG00000059552*
- Return the length of following chromosomes in human and mouse
  - 2
  - 7
  - X
  - Which are the longest?

# Version control

GIT – Track and store revisions/versions of files

- https://try.github.io/

- Help
  `$ git help [<git_command>]`
- Configuration
  `$ git config`

# Version control

GIT – Track and store revisions/versions of files

- Initialize
  `$ git init`
- Show status
  `$ git status`
- Track files
  `$ git add <filename>`
- Commit changes
  `$ git commit [-m “<commit_message>”]`
  `$ git reset`
- Show logs
  `$ git log`
- Checkout a commit
  `$ git checkout <checksum>`
- Show differences between revisions
  `$ git diff [<checksum1> [<checksum2>]]`

# Version control

GIT – Track and store revisions/versions of files

- Branching
  `$ git branch`
  `$ git branch <'new_branch'>`
- Merging
  `$ git merge <new_branch>`
  - Merge conflicts: same file modified on 2 seperate branches
- Delete branch
  `$ git branch -d <new_branch>`
- Remotes
  - GitHub (public repositories)
- Clone repository
  `$ git clone <repository_name> <local_dir>`
- Update repository
  `$ git pull <remote_name> <branch_name>`
- Submit changes
  `$ git push <remote_name> <branch_name>`

# Version control

Exercises

- Create a directory *`db_git`* and copy some of the course files to this new directory
- Create a git repository in this directory
    - Make sure your user_name and user_email are set correctly (HINT: `git config`)
- Commit all `.sql` files and all other files with two different commit messages
    - Check your commit history
- Add a README file to your repository
- Create a second branch in your repository
    - Change to this new branch
    - Add and delete some files
    - Add some lines to your README file
    - Show the differences between your 2 branches
- Include the changes from your new branch into your original branch
- Delete your second branch

# Version control

Exercises

- Go to https://github.com/ and create a new repository *db_github*
- Add the contents of your existing *db_git* repository to your newly created remote one.
- Check the results