Bio4j: bigger, faster, leaner

Pablo Pareja-Tobes, Alexey Alekhin, Evdokim Kovach, Marina Manrique, Eduardo Pareja, Raquel Tobes and Eduardo Pareja-Tobes

April 8, IWBBIO-2014

Bio4j

ohnosequences!

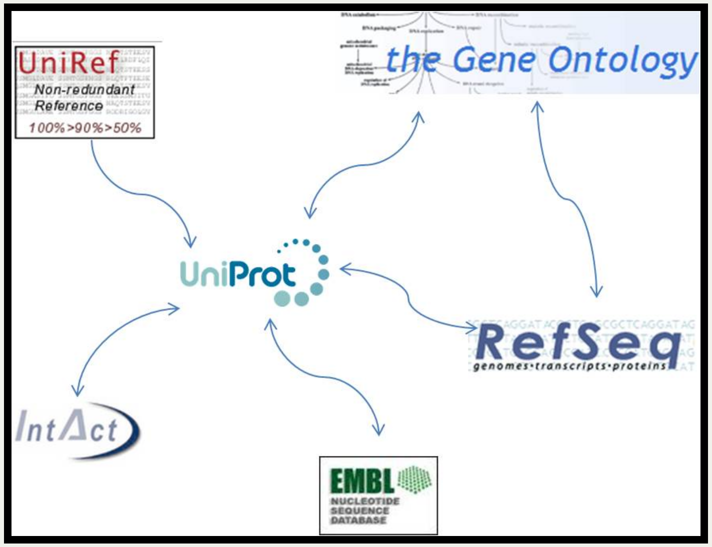
Era7 bioinformatics

Introduction

What is Bio4j?

Bio4j is a bioinformatics *graph*-based data platform **integrating** the most representative **open data sources** around **protein information**

Data sources



- *UniProt KB* (SwissProt + Trembl)
- *Gene Ontology* (GO)
- *UniRef* (50,90,100)
- *RefSeq*
- *NCBI Taxonomy*
- *Expasy Enzyme DB*

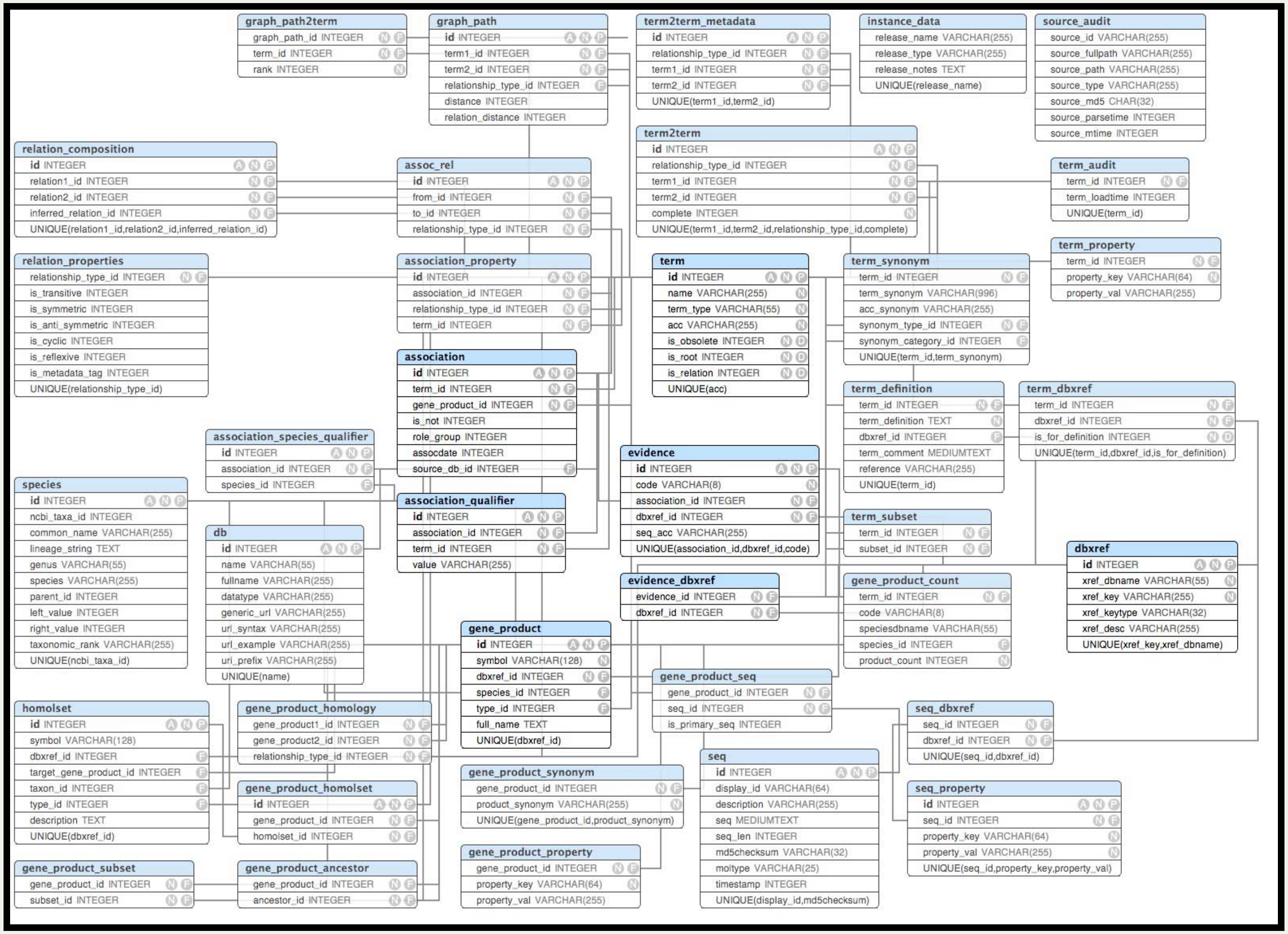
It’s open!

- Code is under the **AGPLv3** license
- Only **Open Data** is integrated
- Implementation & release process is **100% public** and totally transparent

Biology & Databases today

- Highly **interconnected overlapping** knowledge
- spread over **different data sources**
- maintained in the **Relational Databases**
  or sometimes even just as plain **CSV files**

That might be fine for simple scenarios
but as the **amount** and **diversity** of data grows,
domain models become *crazily complicated!*



graph_path2term
graph_path_id INTEGER
term_id INTEGER
rank INTEGER

graph_path
id INTEGER
term1_id INTEGER
term2_id INTEGER
relationship_type_id INTEGER
distance INTEGER
relation_distance INTEGER

term2term_metadata
id INTEGER
relationship_type_id INTEGER
term1_id INTEGER
term2_id INTEGER
UNIQUE(term1_id,term2_id)

instance_data
release_name VARCHAR(255)
release_type VARCHAR(255)
release_notes TEXT
UNIQUE(release_name)

source_audit
source_id VARCHAR(255)
source_fullpath VARCHAR(255)
source_path VARCHAR(255)
source_type VARCHAR(255)
source_md5 CHAR(32)
source_parsetime INTEGER
source_mtime INTEGER

relation_composition
id INTEGER
relation1_id INTEGER
relation2_id INTEGER
inferred_relation_id INTEGER
UNIQUE(relation1_id,relation2_id,inferred_relation_id)

assoc_rel
id INTEGER
from_id INTEGER
to_id INTEGER
relationship_type_id INTEGER

term2term
id INTEGER
relationship_type_id INTEGER
term1_id INTEGER
term2_id INTEGER
complete INTEGER
UNIQUE(term1_id,term2_id,relationship_type_id,complete)

term_audit
term_id INTEGER
term_loadtime INTEGER
UNIQUE(term_id)

term_property
term_id INTEGER
property_key VARCHAR(64)
property_val VARCHAR(255)

relation_properties
relationship_type_id INTEGER
is_transitive INTEGER
is_symmetric INTEGER
is_anti_symmetric INTEGER
is_cyclic INTEGER
is_reflexive INTEGER
is_metadata_tag INTEGER
UNIQUE(relationship_type_id)

association_property
id INTEGER
association_id INTEGER
relationship_type_id INTEGER
term_id INTEGER

term
id INTEGER
name VARCHAR(255)
term_type VARCHAR(55)
acc VARCHAR(255)
is_obsolete INTEGER
is_root INTEGER
is_relation INTEGER
UNIQUE(acc)

term_synonym
term_id INTEGER
term_synonym VARCHAR(996)
acc_synonym VARCHAR(255)
synonym_type_id INTEGER
synonym_category_id INTEGER
UNIQUE(term_id,term_synonym)

association
id INTEGER
term_id INTEGER
gene_product_id INTEGER
is_not INTEGER
role_group INTEGER
assocdate INTEGER
source_db_id INTEGER

term_definition
term_id INTEGER
term_definition TEXT
dbxref_id INTEGER
term_comment MEDIUMTEXT
reference VARCHAR(255)
UNIQUE(term_id)

term_dbxref
term_id INTEGER
dbxref_id INTEGER
is_for_definition INTEGER
UNIQUE(term_id,dbxref_id,is_for_definition)

association_species_qualifier
id INTEGER
association_id INTEGER
species_id INTEGER

evidence
id INTEGER
code VARCHAR(8)
association_id INTEGER
dbxref_id INTEGER
seq_acc VARCHAR(255)
UNIQUE(association_id,dbxref_id,code)

species
id INTEGER
ncbi_taxa_id INTEGER
common_name VARCHAR(255)
lineage_string TEXT
genus VARCHAR(55)
species VARCHAR(255)
parent_id INTEGER
left_value INTEGER
right_value INTEGER
taxonomic_rank VARCHAR(255)
UNIQUE(ncbi_taxa_id)

association_qualifier
id INTEGER
association_id INTEGER
term_id INTEGER
value VARCHAR(255)

db
id INTEGER
name VARCHAR(55)
fullname VARCHAR(255)
datatype VARCHAR(255)
generic_url VARCHAR(255)
url_syntax VARCHAR(255)
url_example VARCHAR(255)
uri_prefix VARCHAR(255)
UNIQUE(name)

term_subset
term_id INTEGER
subset_id INTEGER

dbxref
id INTEGER
xref_dbname VARCHAR(55)
xref_key VARCHAR(255)
xref_keytype VARCHAR(32)
xref_desc VARCHAR(255)
UNIQUE(xref_key,xref_dbname)

evidence_dbxref
evidence_id INTEGER
dbxref_id INTEGER

gene_product_count
term_id INTEGER
code VARCHAR(8)
speciesdbname VARCHAR(55)
species_id INTEGER
product_count INTEGER

gene_product
id INTEGER
symbol VARCHAR(128)
dbxref_id INTEGER
species_id INTEGER
type_id INTEGER
full_name TEXT
UNIQUE(dbxref_id)

gene_product_seq
gene_product_id INTEGER
seq_id INTEGER
is_primary_seq INTEGER

homolset
id INTEGER
symbol VARCHAR(128)
dbxref_id INTEGER
target_gene_product_id INTEGER
taxon_id INTEGER
type_id INTEGER
description TEXT
UNIQUE(dbxref_id)

gene_product_homology
gene_product1_id INTEGER
gene_product2_id INTEGER
relationship_type_id INTEGER

seq_dbxref
seq_id INTEGER
dbxref_id INTEGER
UNIQUE(seq_id,dbxref_id)

seq
id INTEGER
display_id VARCHAR(64)
description VARCHAR(255)
seq MEDIUMTEXT
seq_len INTEGER
md5checksum VARCHAR(32)
moltype VARCHAR(25)
timestamp INTEGER
UNIQUE(display_id,md5checksum)

gene_product_synonym
gene_product_id INTEGER
product_synonym VARCHAR(255)
UNIQUE(gene_product_id,product_synonym)

gene_product_homolset
id INTEGER
gene_product_id INTEGER
homolset_id INTEGER

seq_property
id INTEGER
seq_id INTEGER
property_key VARCHAR(64)
property_val VARCHAR(255)
UNIQUE(seq_id,property_key,property_val)

gene_product_property
gene_product_id INTEGER
property_key VARCHAR(64)
property_val VARCHAR(255)

gene_product_subset
gene_product_id INTEGER
subset_id INTEGER

gene_product_ancestor
gene_product_id INTEGER
ancestor_id INTEGER

Doesn't look very compelling right?

Relational model

With relational paradigm the double implication

Entity $\Leftrightarrow$ **Table**

doesn’t go both ways, which implies

- **auxiliary tables**
- **artificial IDs**
- dealing with **raw tables**
(in spite of entity-relationship diagrams)

Integrating new knowledge becomes **difficult**

Biology $\neq$ Table

- **Life** in general and **biology** in particular are probably not 100% like a graph…
- but one thing is sure: they ***are not a set of tables!***

Why graph databases?

- Data is stored in a way that **semantically represents its own structure**
- Incorporating new data is easy $\Rightarrow$ it's **scalable**
- **Vertex-centric** *(local)* indices allow to overcome the supernode problem

Why in the cloud?

amazon
web services™

Data as a service

- Services interoperability
- Data distribution
- Backup and storage
- Scalability
- Cost-effectiveness

Bio4j

=

Bio Data

+

Graph Databases

+

The Cloud

Details about Bio4j

How it all started

- Need for **massive access** to *Gene Ontology* annotations
- **BG7** bacterial genome annotation system
- Need for massive direct access to **protein information**

More and more data!

- As *other* data sources were becoming a *bottleneck* they were integrated into Bio4j
- First it was Uniprot KB, then Uniref, …
- And **we didn’t stop yet!**

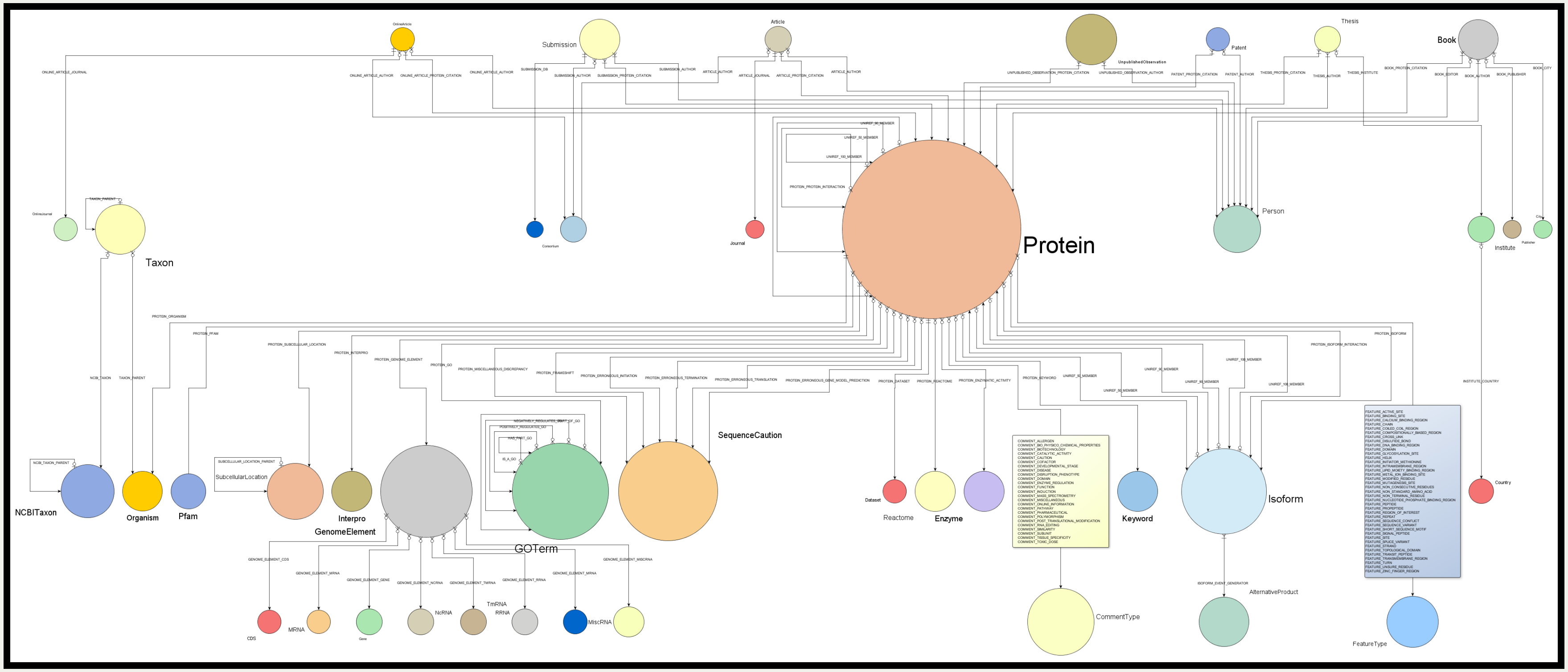
Different layers of Bio4j

1. Abstract **domain model** with precise typing
2. Universal Blueprints implementation
3. **Technology-specific** versions:
    - Neo4j
    - Titan (WIP)
    - OrientDB (planned)

Different **graph topologies** at the storage level,
same **domain model** in the client's code

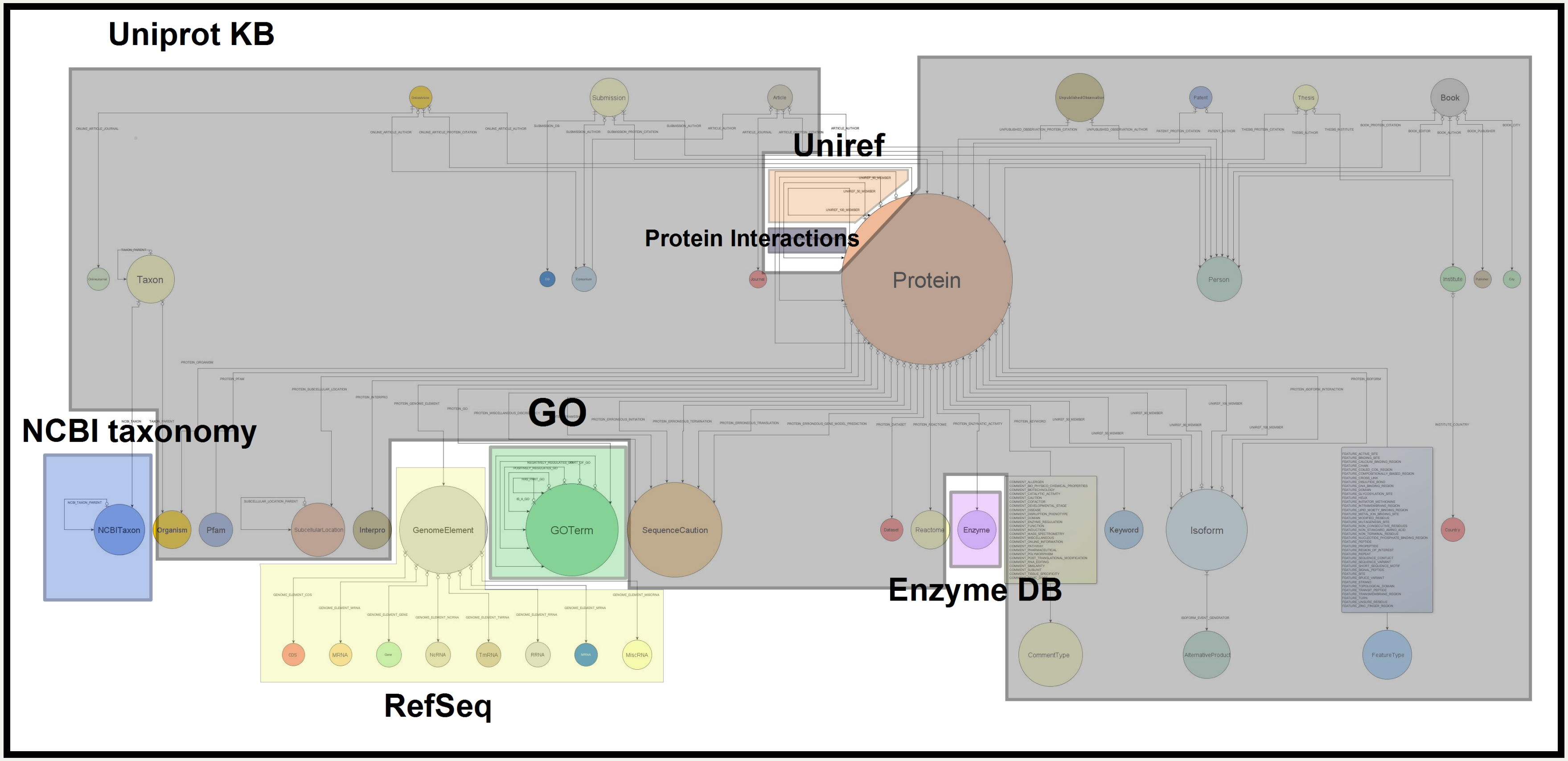
Bio4j domain model

- 10^9 edges of **150 types**
- 2×10^8 nodes of **40 types**
- 6×10^8 properties



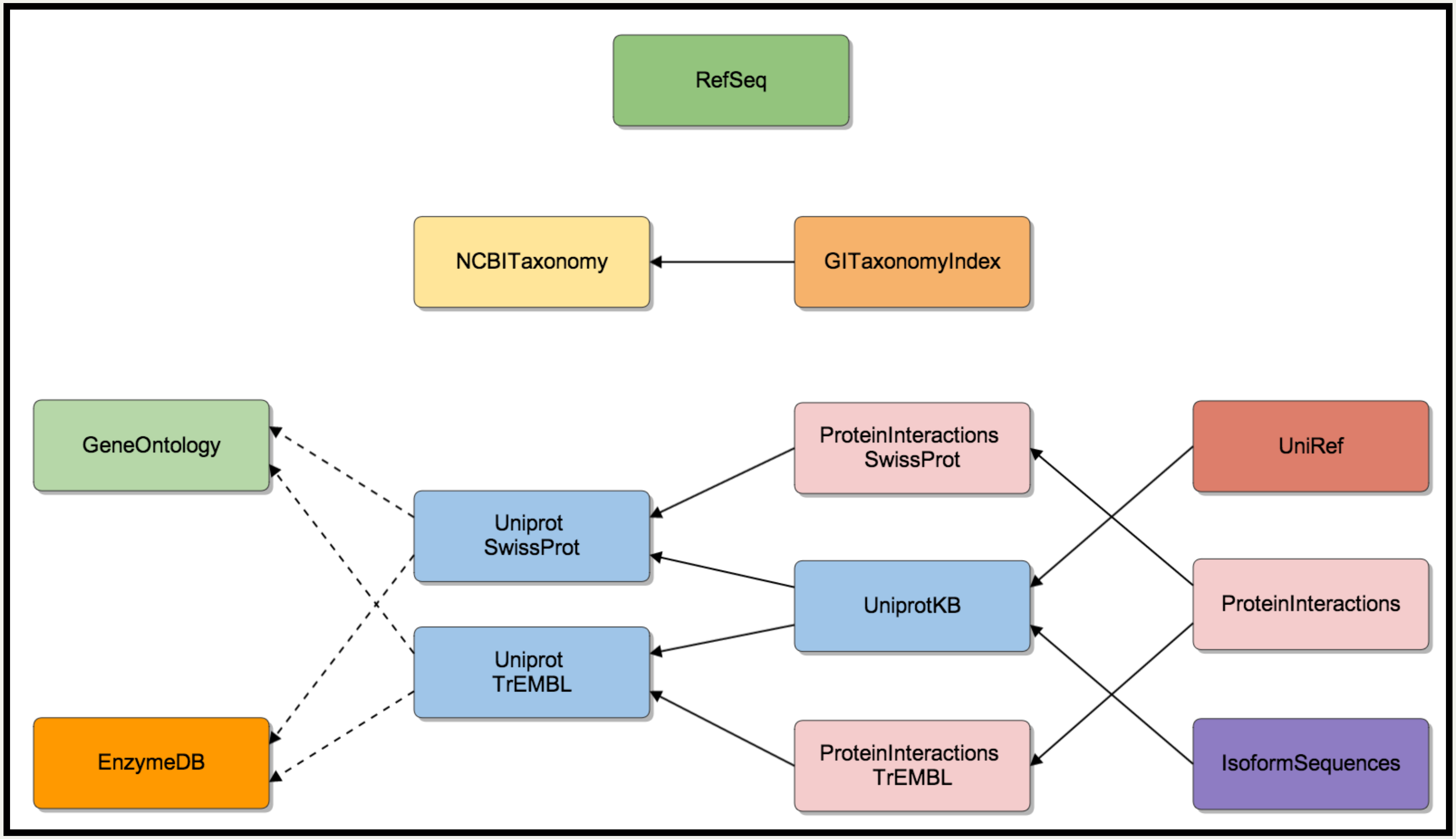
Bio4j structure

The importing process is **modular** and **customizable** allowing you to import just the data you are interested in



Bio4j module system

Statika helps to manage dependencies between modules and simplifies import and deployment in the cloud



Under the hood

How we use Bio4j in Era7

- **BG7** genome annotation
- **MG7** metagenomics analysis
- Comparative genomics, network analysis, genome assembly, …

How others use Bio4j

Ohio State University

- **Integration** and **analysis** of Chip-seq data
- **Modeling** genomic information and **gene regulatory networks**

Berkeley Phylogenomics Group

- Graph database for *Big Data challenges* in **genomics** developed **on top of Bio4j**

How we develop Bio4j

- Java + Scala source code
- Statika-based module system
- SBT for building sources
  and automated tests & release
- Git + Github: versioning, docs,
  collaboration, coordination

Who's doing Bio4j

Ohnosequences!

Era7 bioinformatics R&D group

Pablo Pareja	project leader & main developer
Eduardo Pareja-Tobes	technology & architecture
Raquel Tobes	bio data integration
Marina Manrique	bio data integration
Alexey Alekhin	module system developer
Evdokim Kovach	developer

Contacts

- @bio4j **Twitter** for news
- bio4j **Github** org for the development process
- bio4j-user **Google group** for the user feedback
- bio4j **Linkedin**

bio4j.com

Thank you for attention!

The source and the latest version of these slides can be found at

`github.com/ohnosequences/IWBBIO-2014`