What is BioSQL?

- Interoperable persistence layer (relational data store) for the Bio* projects
  - Bindings for BioPerl, Biojava, Biopython, BioRuby
- Very flexible, highly normalized, weakly-typed schema
  - Focal entities are Bioentry, Seqfeature, Term
  - Encompasses sequences, sequence features, annotation, and ontology terms
- Open-source, open development project
- Very stable since 4 years
- Actively used (Google Scholar: 72 matches)
BioSQL History

- Ewan Birney started BioSQL in Nov 2001
  - Initial use-case was to have a replacement for SRS supporting BioPerl sequence objects
- Schema redesigned at the 2002 Biohackathons in Tucson and Cape Town
- Overhaul at the 2003 Biohackathon in Singapore
  - Taxon model resembling NCBI’s
  - Full ontology model resembling GO’s model
- Added Phylodb module at 2006 Phylo-Hackathon
  - Phylogenetic tree model, query optimizations
BioSQL Use-Cases:
Persistence API for Bio*

- Persistent storage for my favorite Bio* toolkit
  - Relational model accommodates object model
  - Persistence API with transparent insert, update, delete
BioSQL Use-Cases:
Standard DB for Seqs & Trees

- ‘Local GenBank/UniProt/TreeBASE with random access’
  - Local cache or replication of public databanks
  - Indexed random access, easy retrieval
- ‘GenBank/UniProt/TreeBASE in relational format’
  - Normalized schema, predictably populated
  - Allows arbitrary queries
- ‘Database for my lab data’
  - Store sequences, genes, trees, annotation, …
  - Add, update, modify, remove
BioSQL Use-Cases: Data Integration

- ‘Integrate GenBank, UniProt, UniGene, Entrez Gene, NCBI Taxonomy, ITIS, IPNI, ToL, TreeBASE, …’
  - Unifying relational schema
  - Provide common (abstracted) view on different sources of data and annotation
BioSQL: Weakly typed model

- Very flexible, extensible
- Ontology-encoded types
- Explicit semantics
- Semantic web-friendly, maps easily to RDF
BioSQL: Phylodb Module
use Bio::Seq; use Bio::SeqFeature::Generic;

# retrieve the sequence object somehow ...
$adp = $db->get_object_adaptor("Bio::SeqI");
$dbseq = $adp->find_by_unique_key(Bio::Seq->new(-accession_number => "NM_000149", -namespace => "RefSeq"));

# create a feature as new annotation
$feat = Bio::SeqFeature::Generic->new(-primary_tag => "TFBS", -source_tag => "My Lab", -start=>23,-end=>27,-strand=>-1);

# add new annotation to the sequence
$dbseq->add_SeqFeature($feat);

# update in the database
$dbseq->store();
Phyloinformatics Initiative

- Hundreds of evolutionary analysis tools
  - Inference of phylogenetic trees, evolutionary model comparison, hypothesis testing in comparative frameworks
  - Algorithms using pairwise distance, maximum parsimony, maximum likelihood, bayesian probability
  - Models for substitution, selection, migration, recombination, ...

- But little interoperability
  - No infrastructure for seamless workflow integration:
    - Incompatible program interfaces
    - Custom, incompatible data exchange formats
    - Standards that do exist are insuffuciently adhered to
  - No infrastructure for service-oriented or semantic web

Phyloinformatics Initiative

- Hundreds of evolutionary analysis tools
  - Inference of phylogenetic trees, evolutionary model comparison, hypothesis testing in comparative frameworks
  - Algorithms using pairwise distance, maximum parsimony, maximum likelihood, Bayesian probability
  - Models for substitution, selection, migration, recombination, ...

Our goal is to address and remedy these issues.

But little interoperability
- No infrastructure for seamless workflow integration:
  - Incompatible program interfaces
  - Custom, incompatible data exchange formats
  - Standards that do exist are insufficiently adhered to
- No infrastructure for service-oriented or semantic web
Held at NESCent on Dec 11-15, 2006

Objectives:
- Lower the barrier to using and combining the existing sophisticated analysis tools
- Interoperability between tools by glue code and by removing obstacles to data exchange
BioSQL Targets

1. Develop a relational model for phylogenetic trees in BioSQL that can be used as an optional add-on. The model should accomplish the following basic features.
   - Stores gene trees as well as species trees.
   - Accommodates binary trees, polytomies, and anastomizing trees.
   - Allows attributes with values to be assigned to edges between nodes, such as branch length, support value(s), or posterior probability.
   - Allows attributes with values to be assigned to nodes in the tree, such as duplication/speciation events, calibrated or estimated time, etc.
   - Allows the results of precomputes for accelerating topological queries to be stored.

2. Script to load trees into the schema from, e.g., NEXUS files.

3. Script to run the precompute calculations
   - Nested Set values (left and right values)
   - Transitive closure
   - Path through nodes

4. Define SQL queries to resolve typical topological queries against a database of trees, resulting from, e.g., a bootstrap analysis of a phylogeny, or conflicting gene and species trees. This relates to use cases Reconcile trees (3.19), Determine concordance between two or more phylogenies (3.15), Infer tree (3.12) and Calculate support values (3.13).
   1. Find the LCA (least common ancestor) of nodes A and B
   2. Find the oldest ancestor node of A such that B is not descended from the ancestor
   3. Find the subtree rooted at LCA(A,B) of nodes A and B (minimal spanning clade)
   4. Find the maximum spanning clade that includes nodes A and B but not C (atom query)
   5. Tree pattern match - all trees that have the same topology between chosen set of taxa (or genes)
      - all trees for which the minimum spanning clade of nodes A and B includes node C (as identified by label)
      - all trees for which the minimum spanning clade of nodes A and B does not include node C (as identified by label)
   6. Subsetting trees:
      - all trees that have at least the given nodes, identified by label
   7. Tree projection - obtain the subtree induced by the chosen set of nodes

5. Create simple web-based user interface to browse and search the database of trees, including topological queries.

In collaboration with the Bioperl group (Aaron Mackey) we also devised a draft mapping of alignment objects as well as character state matrix data into BioSQL.

Todos:
1. Implement the tree projection query.
Topology query support

- Query optimization precomputes
  - Nested-set enumeration
  - Transitive closure and node path
- Topology queries in SQL
Eleven students (each with at least one expert mentor) worked remotely on open-source phyloinformatic software.

Jamie Estill: Command-line tools for BioSQL

Greg Jordan (mentor: Bill Piel): Web-based tree visualization & editing
Goal: Deployable Tree Database
Summary

- BioSQL is a very flexible, weakly-typed ontology-driven, stable relational schema to capture richly annotated databank entries.
- BioSQL is supported as the persistent storage across the Bio* projects with easy-to-use language bindings.
- There are a variety of powerful scripts for loading & updating sequence annotation, ontologies, taxonomies, and trees.
- URLs:
  - BioSQL: http://www.biosql.org
  - Hackathon: http://hackathon.nescent.org
  - Summer of Code: http://phylosoc.nescent.org
Acknowledgements

- Phyloinformatics & Summer of Code Hackathon participants
  - Jamie Estill, Bill Piel, Gregory Jordan, & others
- NESCent residents & leadership
  - T. Vision, K. Smith, and various postdocs
- Bio* contributors and core developers
- Hackathon Sponsors
  - NESCent (Durham 2006), Apple (Singapore 2003), Electric Genetics (Cape Town), O’Reilly (Tucson 2002)