CRAWL
Chado Restful Access Webservice Layer

a programmatic interface for querying pathogen genomics data

Giles Velarde, Pathogen Genomics
VAPORWARE ALERT!

- Rapid prototypes
- Proofs of concept
- Requirements gathering

- Conditions apply
THE PATH

• Chado
• GeneDB
• Cooperation with EupathDB
• Web services & APIs
• The Way of the CRAWL
Chado is a relational database schema that underlies many GMOD installations. It is capable of representing many of the general classes of data frequently encountered in modern biology such as sequence, sequence comparisons, phenotypes, genotypes, ontologies, publications, and phylogeny. It has been designed to handle complex representations of biological knowledge and should be considered one of the most sophisticated relational schemas currently available in molecular biology. The price of this capability is that the new user must spend some time becoming familiar with its fundamentals.

- A database for very deep curation
- An integrated database
- A database that is generic enough to use for any organism
Pathogen DB - Unique Resource!

- Built on Chado
- 45 organisms
  - Apicomplexan Protozoa
  - Kinetoplastid Protozoa
  - Parasitic Helminths
  - Bacteria
- Cross-organism computed data
  - Orthologues
  - Domains
• Pathogen DB web front-end
  - Hibernate
    • DAO caches
    • Lucene index search
• Weekly data updates
**EUPATHDB COOPERATION**

- **EupathDB**
  - Functional genomics integrative resources
- **Collaboration**
  - Annotation team
    - Sanger
    - Seattle SBRI
    - UGA
  - TriTrypDB / PlasmoDB
    - Data integration
      - GeneDB genomics data
      - Functional genomics data sets
THE CHALLENGE

• EupathDB
  - Need to know what has changed
  - Need to be able to get the data

• Remote annotation
  • DB-Artemis via VPN
  • Consistency
  - Need to build Rich Internet Applications

• Sanger
  - Need to exploit our own data as well!
  - Chado-complexity - SQL hard
THE NEEDS

• Rapid prototype
  - Quick to implement new queries
  - Must run directly off DB
    • No time to rebuild caches
    • Pure SQL
• Lightweight
  - Must not tax existing website
    • Bioinformatics PhD students...
  - JS,CURL/WGET, PERL, R...
    • REST
• Respond to user’s specific needs
THE TECHNOLOGY

- Python, Jython
- CherryPy
  - Multi-threaded web app server
  - Simple to reuse controller classes in different contexts
- Ropy
  - SQL files
  - XML/JSON
• Library
  – Unit testing...

• Standalone app server
  – MVC ... there is no V
  – enforced decoupling of the data layer from the view

• Command line app
  – Direct DB access
    • WS not suitable for LSF jobs
  – PERL wrapper module
THE PURPOSE

Mask the complexity of the SQL as much as possible, & allow you to get on with data analysis & development.
USE CASES
WHAT'S NEW WEB-SERVICE

- **EupathDB**
  - Queries our services daily
  - recent annotation changes
  - Displays on their gene page a link to GeneDB
EXPLOITING PATHOGENS DB LOCALLY

Using the PERL command-line wrapper
WHAT'S NEW

(FLORA LOGAN, ULRIKE BÖHME & MATT ROGERS)

```perl
use strict;
use warnings;
use Crawl::Crawler;

my $crawler = Crawl::Crawler->new();

# Store the annotation_changes in a temporary hashtable (returned by crawler as a collection).
my %annotation_changes;
foreach my %annotation_change (@{$crawler})
  ->query("histories/annotation_changes", {taxonomyID => "185431", since => "2009-01-01"})
  ->{response->{results}}
  { $annotation_changes{$_} = $annotation_change; }

# Check the genes in the chromosome you're interested in to see which ones have changed.
print "gene\tfeature\tfeature_type\tuser\tdate\tchange\n";
foreach my %gene (@{$crawler})
  ->query("genes/inregion", {region => "Tb927_09_v4"})
  ->{response->{genes}}
  {
    my $gene_name = %gene->{gene};
    if (exists($annotation_changes->{$gene_name})) {
      my $gene_result = $annotation_changes->{$gene_name};
      print $gene_result->{gene} . "\t";
      print $gene_result->{feature} . "\t";
      print $gene_result->{feature_type} . "\t";
      print $gene_result->{type} . "\t";
      print $gene_result->{user} . "\t";
      print $gene_result->{date} . "\t";
      print $gene_result->{change} . "\n";
    }
  }
```

PFAM DOMAINS -> CODA

(ADAM REID)
A query that is useful for a biologist locally could well be useful for a bioinformatician remotely.
COMMAND LINE EXAMPLES
COMMAND LINE - GET ORGANISMS

Uses SpiderMonkey, Jsawk & JSONQuery
**Command Line - Format Genes**

```
$ crawler.py -query genes/inregion -database localhost/pathogens?pathdb \
    -region Pf3D7_01 \n    | jsawk "return this.response.genes" \n    | jsawk -a 'return this.join("\n")' "return this.gene + \t + this.fmin + \t + this.fmax"
```

Password:

<table>
<thead>
<tr>
<th>Gene</th>
<th>Fmin</th>
<th>Fmax</th>
</tr>
</thead>
<tbody>
<tr>
<td>PFA0170c</td>
<td>148148</td>
<td>153011</td>
</tr>
<tr>
<td>PFA0315W</td>
<td>272692</td>
<td>273641</td>
</tr>
<tr>
<td>PFA0380w</td>
<td>315773</td>
<td>320559</td>
</tr>
<tr>
<td>PFA0440W</td>
<td>364470</td>
<td>365145</td>
</tr>
<tr>
<td>PFA0515W</td>
<td>406892</td>
<td>413323</td>
</tr>
<tr>
<td>PFA0370w</td>
<td>304090</td>
<td>304552</td>
</tr>
<tr>
<td>PFA0045c</td>
<td>62419</td>
<td>63633</td>
</tr>
<tr>
<td>PFA0115W</td>
<td>104935</td>
<td>105441</td>
</tr>
<tr>
<td>PFA0560c</td>
<td>443694</td>
<td>444441</td>
</tr>
<tr>
<td>PFA0310c</td>
<td>265446</td>
<td>269412</td>
</tr>
<tr>
<td>PFA0240W</td>
<td>211107</td>
<td>213243</td>
</tr>
<tr>
<td>PFA0025c</td>
<td>53391</td>
<td>53503</td>
</tr>
<tr>
<td>PFA0570w</td>
<td>447176</td>
<td>450433</td>
</tr>
<tr>
<td>PFA0705c</td>
<td>562298</td>
<td>563256</td>
</tr>
<tr>
<td>PFA0335W</td>
<td>287166</td>
<td>289134</td>
</tr>
</tbody>
</table>
```
COLLABORATIVE INTERFACES

Rich internet applications
AJAX TECHNIQUES

- Data refresh without page refresh
- Conservative
  - Autocomplete
- Advanced
  - Rich internet applications
e.g. Google maps

Fundamentally depends on web services
REMINDER:
VAPORWARE ALERT!

Very early prototypes...
Anything can go wrong!
A PROTOTYPE WEB-APP

(TIM CARVER)

Good example of an AJAX driven application.
Some people call it Web-Artemis.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Feature Start</th>
<th>Feature End</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>PFA0480w</td>
<td>gene</td>
<td>381850</td>
<td>383578</td>
<td></td>
</tr>
<tr>
<td>PFA0480w.mRNA</td>
<td>mRNA</td>
<td>381850</td>
<td>383578</td>
<td></td>
</tr>
<tr>
<td>PFA0480w:exon:1</td>
<td>exon</td>
<td>381850</td>
<td>383578</td>
<td></td>
</tr>
<tr>
<td>PFA0480w.pep</td>
<td>polypeptide</td>
<td>381850</td>
<td>383578</td>
<td></td>
</tr>
<tr>
<td>PFA0485w</td>
<td>gene</td>
<td>384437</td>
<td>388779</td>
<td></td>
</tr>
<tr>
<td>PFA0485w.2</td>
<td>mRNA</td>
<td>384437</td>
<td>388779</td>
<td></td>
</tr>
<tr>
<td>PFA0485w.2:exon:2</td>
<td>exon</td>
<td>385040</td>
<td>385166</td>
<td></td>
</tr>
<tr>
<td>PFA0485w.2:exon:1</td>
<td>exon</td>
<td>384437</td>
<td>384876</td>
<td></td>
</tr>
<tr>
<td>PFA0485w.2:exon:18</td>
<td>exon</td>
<td>388749</td>
<td>388779</td>
<td></td>
</tr>
<tr>
<td>PFA0485w.2:exon:3</td>
<td>exon</td>
<td>385353</td>
<td>385424</td>
<td></td>
</tr>
<tr>
<td>PFA0485w.2:exon:4</td>
<td>exon</td>
<td>385553</td>
<td>385692</td>
<td></td>
</tr>
<tr>
<td>PFA0485w.2:exon:5</td>
<td>exon</td>
<td>385780</td>
<td>385961</td>
<td></td>
</tr>
<tr>
<td>PFA0485w.2:exon:6</td>
<td>exon</td>
<td>386050</td>
<td>386253</td>
<td></td>
</tr>
<tr>
<td>PFA0485w.2:exon:7</td>
<td>exon</td>
<td>386417</td>
<td>386539</td>
<td></td>
</tr>
<tr>
<td>PFA0485w.2:pep</td>
<td>polypeptide</td>
<td>384437</td>
<td>388779</td>
<td>comment=alternate spliced form revealed by splice bridging read pair</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>transcriptome sequence. Confirmed by 4 reads (PMID:20141604).</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Feature Start</td>
<td>Feature End</td>
<td>Properties</td>
</tr>
<tr>
<td>----------------</td>
<td>---------</td>
<td>---------------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PFA0485w</td>
<td>gene</td>
<td>384437</td>
<td>389094</td>
<td></td>
</tr>
<tr>
<td>PFA0485w.2</td>
<td>mRNA</td>
<td>384437</td>
<td>388779</td>
<td></td>
</tr>
<tr>
<td>PFA0485w.2:exon:6</td>
<td>exon</td>
<td>386050</td>
<td>386253</td>
<td></td>
</tr>
<tr>
<td>PFA0485w.2:exon:11</td>
<td>exon</td>
<td>387242</td>
<td>387304</td>
<td></td>
</tr>
<tr>
<td>PFA0485w.2:exon:12</td>
<td>exon</td>
<td>387484</td>
<td>387547</td>
<td></td>
</tr>
<tr>
<td>PFA0485w.2:pep</td>
<td>polypeptide</td>
<td>384437</td>
<td>388779</td>
<td>comment=alternate spliced form revealed by splice bridging read pair transcriptome sequence. Confirmed by 4 reads (PMID:20141604):</td>
</tr>
<tr>
<td>PFA0485w.2:exon:13</td>
<td>exon</td>
<td>387711</td>
<td>387760</td>
<td></td>
</tr>
<tr>
<td>PFA0485w.2:exon:14</td>
<td>exon</td>
<td>387844</td>
<td>387876</td>
<td></td>
</tr>
<tr>
<td>PFA0485w.2:exon:15</td>
<td>exon</td>
<td>388024</td>
<td>388196</td>
<td></td>
</tr>
<tr>
<td>PFA0485w.2:exon:16</td>
<td>exon</td>
<td>388367</td>
<td>388453</td>
<td></td>
</tr>
<tr>
<td>PFA0485w.2:exon:1</td>
<td>exon</td>
<td>384437</td>
<td>384876</td>
<td></td>
</tr>
<tr>
<td>PFA0485w.2:exon:2</td>
<td>exon</td>
<td>385040</td>
<td>385166</td>
<td></td>
</tr>
<tr>
<td>PFA0485w.2:exon:3</td>
<td>exon</td>
<td>385315</td>
<td>385424</td>
<td></td>
</tr>
<tr>
<td>PFA0485w.2:exon:4</td>
<td>exon</td>
<td>385553</td>
<td>385692</td>
<td></td>
</tr>
<tr>
<td>PFA0485w.2:exon:5</td>
<td>exon</td>
<td>385780</td>
<td>385961</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Feature Start</td>
<td>Feature End</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>---------</td>
<td>---------------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>PFA0485w</td>
<td>gene</td>
<td>384437</td>
<td>389046</td>
<td></td>
</tr>
<tr>
<td>PFA0485w.2</td>
<td>mRNA</td>
<td>384437</td>
<td>388779</td>
<td></td>
</tr>
<tr>
<td>PFA0485w.2:exon:6</td>
<td>exon</td>
<td>386050</td>
<td>386253</td>
<td></td>
</tr>
<tr>
<td>PFA0485w.2:exon:11</td>
<td>exon</td>
<td>387242</td>
<td>387304</td>
<td></td>
</tr>
<tr>
<td>PFA0485w.2:exon:12</td>
<td>exon</td>
<td>387484</td>
<td>387547</td>
<td></td>
</tr>
<tr>
<td>PFA0485w.2:pep</td>
<td>polypeptide</td>
<td>384437</td>
<td>388779</td>
<td></td>
</tr>
</tbody>
</table>

**Product:** phosphatidate cytidylyltransferase, putative

**Gene Ontology:**
- GO:0020011; aspect:C; spiroplast; evidence:inferred from Reviewed Computational Analysis; PMID:11738614
- GO:0016272; aspect:F; transferase activity, transferring phosphorus-containing groups; autocomment:From protein; evidence:Inferred from Electronic Annotation; date=20100113;
- GO:0016020; aspect:G; membrane; autocomment:From protein; evidence:inferred from Electronic Annotation; date=20100113;

**Clusters:**
- PfANKA_02037; pep
- PfANKA_02021; pep
- PFA0485w.1; pep
- PFA0485w.2:pep
POTENTIAL USES

• Genomics visualisation tool
  – Including GeneDB of course
• Community annotation
  – Users won’t have to use a desktop app via VPN
    • Simple annotations
• and ...
SNP-MASHUPS
(TIM CARVER & DAVID AANENSEN)
SMASHUPS! (SNP-MASHUPS)

- Embedding as a widget
  - Combining
    - web services
    - independently written widgets
  - Integrate software that integrates data
The Take Home Message

- The Way of the CRAWL
  - Built as a library first
  - Deployed as
    - Standalone Web services app
    - Command line app
  - Used for
    - Collaborating with EupathDB
    - Query multi-organism data sets in house
      - without going through WS
    - Building RIAs and (s)mashups
GMODREST

• Currently
  - **Python** REST framework (Ropy)
  - Speaks **Chado**

• Time to implement the GMODREST interface?

• Caveats
  - GeneDB’s Chado may have little differences
    • must test on other DBs
<table>
<thead>
<tr>
<th>Adrian Tivey</th>
<th>Adam Reid</th>
<th>Brian Brunk</th>
<th>Christiane Hertz-Fowler</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anne Pajon</td>
<td>Flora Logan</td>
<td>Cary Pennington</td>
<td>Debbie Smith</td>
</tr>
<tr>
<td>Jacqueline McQuillan</td>
<td>Gary Dillon</td>
<td>Cristina Aurrecoechea</td>
<td>Mark Carrington</td>
</tr>
<tr>
<td>Martin Aslett</td>
<td>Matt Holden</td>
<td>John Iodice</td>
<td>Matt Berriman</td>
</tr>
<tr>
<td>Nishadi De Silva</td>
<td>Matt Rogers</td>
<td>Omar Harb</td>
<td>David Roos</td>
</tr>
<tr>
<td>Raeece Naem</td>
<td>Thomas Dan Otto</td>
<td>Steve Fischer</td>
<td></td>
</tr>
<tr>
<td>Robin Houston</td>
<td>Ulrike Böhme</td>
<td>Vishal Nayak</td>
<td></td>
</tr>
<tr>
<td>Tim Carver</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tina Eyre</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>