Bio::DB::Das::Chado

As Middleware?

Scott Cain

Cold Spring Harbor Laboratory

Create the database

```
$ perl Makefile.PL
$ make
$ sudo make install
$ make load_schema
$ make prepdb            # now with Xenopus!
$ make ontologies        # load rel, SO, featureprop
```

Then load some data...

Create some GFF from the spec

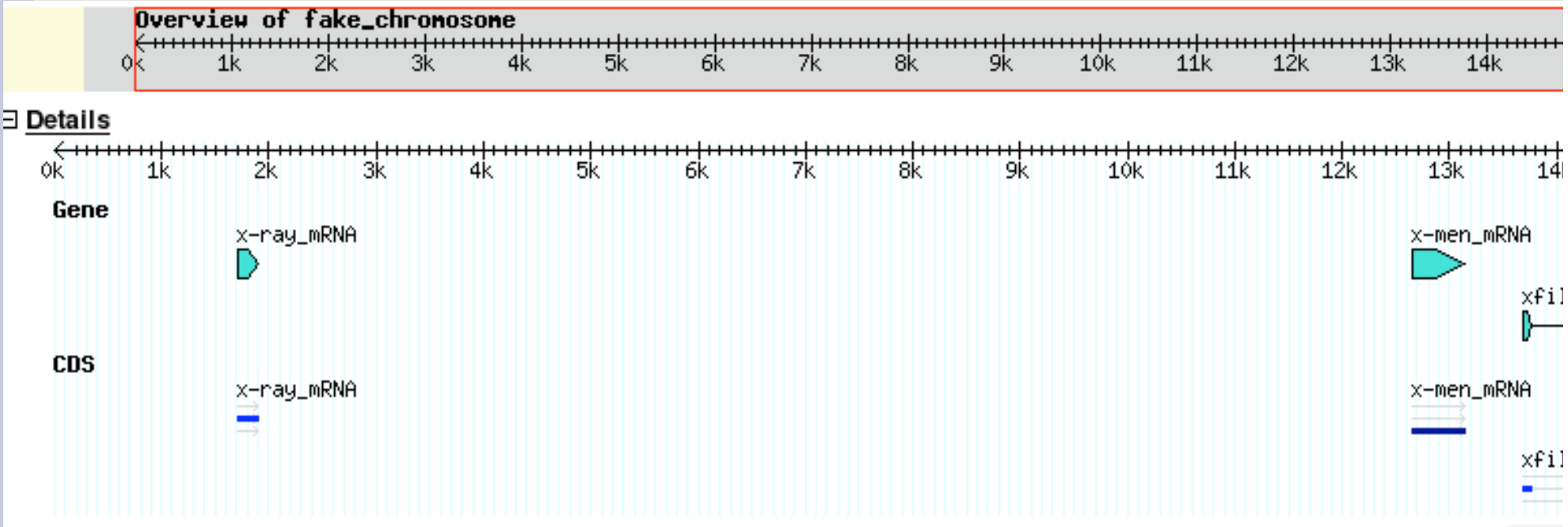
```
fake_chromosome example chromosome 1  15017 .  .  . ID=fake_chromosome;
                                                    Name=fake_chromosome
fake_chromosome example gene   13691  14720 .  +  . ID=xfile;Name=xfile;
                                                    Alias=mulder,scully;
                                                   Note=A test gene for GMOD m
fake_chromosome example mRNA   13691  14720 .  +  . ID=xfile_mRNA;
                                                    Parent=xfile
fake_chromosome example exon   13691  13767 .  +  . Parent=xfile_mRNA
fake_chromosome example exon   14687  14720 .  +  . Parent=xfile_mRNA
fake_chromosome example gene   12648  13136 .  +  . ID=x-men
```

And load it with the bulk loader:

```
$ gmod_bulk_load_gff3.pl -g sample.gff
```

...lots of output...

For kicks, set up GBrowse



Name:	**xfile**
Class:	gene:example
Type:	gene
Source:	example
Position:	fake_chromosome:13691..14720 (+ strand)
Length:	1030
Note:	A test gene for GMOD meeting
dbxref:	GFF_source:example
synonym:	mulder
	scully
	xfile

Parts:

Type:	mRNA
Source:	example
Position:	fake_chromosome:13691..14720 (+ strand)
Length:	1030
dbxref:	GFF_source:example
synonym:	xfile_mRNA

Parts:

Type:	exon
Source:	example
Position:	fake_chromosome:13691..13767 (+ strand)
Length:	77
dbxref:	GFF_source:example
synonym:	auto4
	exon-auto4

Type:	exon
Source:	example
Position:	fake_chromosome:14687..14720 (+ strand)
Length:	34
dbxref:	GFF_source:example
synonym:	auto5
	exon-auto5

```
>xfile_mRNA class=mRNA:example position=fake_chromosome:13691..14720 (+ strand)
atggcgttag tattcatggt tactggtttc gctactgata tcacccagcg tgtaggctgt ggaatcgaac actggtagta
```

Adaptor components

- Bio::DB::Das::Chado
  - Database connection object
- Bio::DB::Das::Chado::Segment
  - Object for any range of DNA
- Bio::DB::Das::Chado::Segment::Feature
  - Feature object

Use Bio::DB::Das::Chado

```
use Bio::DB::Das::Chado;

my $chado = Bio::DB::Das::Chado->new(
      -dsn => "dbi:Pg:dbname=test",
      -user=> "scott",
      -pass=> "" ) || die "no new chado";

my $gene_name = 'xfile';

my ($gene_fo) = $chado->get_features_by_name($gene_name);
```

Use some accessors

```
print "symbol: "       .$gene_fo->display_name."\n";
print "synonyms: "     .join(', ',$gene_fo->synonyms)."\n";
print "description: " .$gene_fo->notes."\n";
print "type: "         .$gene_fo->type."\n";

my ($mRNA) = $gene_fo->sub_SeqFeature();
my @exons  = $mRNA->sub_SeqFeature();

for my $exon (@exons) {
    next unless ($exon->type->method eq 'exon');
    $exon_count++;
    print "exon$exon_count start: ".$exon->start."\n";
    print "exon$exon_count end: "  .$exon->end.  "\n";
    $cds_seq .= $exon->seq->seq; #the first seq
}                                #returns a Bio::Seq objec
```

'Bulk' Output

```
my $gene_name = 'x-*';

my @genes = $chado->get_features_by_name(
                -name => $gene_name,
                -class=> 'gene' );

for my $gene (@genes) {
    print join("\t",
               $gene->feature_id,
               $gene->display_name,
               $gene->organism),"\n";
}
```

Advantages

- Comes 'for free' with GBrowse
- Uses 'familiar' BioPerl idioms, very similar to widely used Bio::DB::GFF (though with fewer methods).

Limitations

- No ability to write
- Incomplete implementation of Bio::DasI; just enough to make GBrowse work
- As an aside: found two bugs while working on this presentation (now fixed in cvs).
- Also, despite the name, has never been tested with a das server.

Conclusion

- Not suitable as a 'general' middleware layer
- However, it may be suitable for some applications, particularly if they are somehow similar to GBrowse or other uses of Bio::DB::GFF.