Machine access to StrepDB data

Govind Chandra <govind.chandra@jic.ac.uk>

December 23, 2011

1 Introduction

This document describes how you may access the data behind StrepDB from your scripts and programs. All you need is some familiarity with SQL and a knowledge of structures of the tables behind StrepDB. Since StrepDB provides access to information about more than one species of *Streptomyces*, the organization of database is different from what it used to be for ScoDB.

The features of all species and all their DNA molecules are now stored in one table named *features*. In this table, accession and serial make the primary key i.e. no two features will have the same *accession* and *serial*. I will refer to this combination of *accession* and *serial* as *accser*. Information about proteins (from all species) is in the table *proteins*. In this table also, *accession* and *serial* make up the primary key. Of course, this table contains only those *accsers* which are proteins. The link between *features* and *proteins* are the *accser*s. To make querying simpler some views have been made. So the view *vprot* has columns both from features and proteins tables. For the purpose of querying views behave exactly as tables.

2 The URL

The URL you need to send your queries to is:
`http://strepdb.streptomyces.org.uk/cgi-bin/strepapi.pl`
See section 4 for an example of how to send queries to this URL to retrieve information from StrepDB.

3 Relevant tables and views

3.1 Tables *features* and *proteins*

```
      Table "public.features"
   Column    |   Type   | Modifiers
-------------+----------+-----------
 id          | text     |
 pri_tag     | text     |
 accession   | text     | not null
 serial      | bigint   | not null
 start_pos   | integer  |
 end_pos     | integer  |
 strand      | smallint |
 annotation  | text     |
 product     | text     |
 db_xref     | text     |
     Table "public.proteins"
  Column   |  Type   | Modifiers
-----------+---------+-----------
 accession | text    | not null
 serial    | bigint  | not null
 gene      | text    |
 aa_seq    | text    |
 nt_seq    | text    |
 pi        | real    |
 mol_wt    | integer |
```

Although *features* and *proteins* are described above, you will probably not need to query them directly. Instead you should be able to get all information you want by querying the view *vprot* described below.

3.2 View *vprot*

This view combines some columns from the *features* and *proteins* tables.

```
        View "public.vprot"
   Column   |   Type   | Modifiers
------------+----------+-----------
 id         | text     |
 pri_tag    | text     |
 accession  | text     |
 serial     | bigint   |
 start_pos  | integer  |
 end_pos    | integer  |
 strand     | smallint |
 annotation | text     |
 product    | text     |
 db_xref    | text     |
 aa_seq     | text     |
 nt_seq     | text     |
 gene       | text     |
 pi         | real     |
 mol_wt     | integer  |
```

3.3 Gene names

The *gene* column of *vprot* contains gene names. At least for some species, this column contains gene identifiers (*id*) as well. The only was to deal with this is to get all the *gene*s and extract out canonical gene names using another script.

Some gene names, which have been suggested by users of StrepDB are in the view *vnames*.

```
      View "public.vnames"
  Column   |  Type  | Modifiers
-----------+--------+-----------
 accession | text   |
 serial    | bigint |
 id        | text   |
 oname     | text   |
 gene      | text   |
 reference | text   |
 comment    | text    |
 email      | text    |
```

This view can be queried as usual. *oname* is the column which contains the suggested gene name.

3.4 Accessions

Each DNA molecule of each species has a different accession number. This, and other information, are stored in the table *organisms*. There is no view for this table. You can query it directly.

```
   Table "public.organisms"
  Column   | Type | Modifiers
-----------+------+-----------
 name      | text |
 accession | text | not null
 file      | text |
 pre       | text |
 taxonomy  | text |
 molecule  | text |
 tla       | text |


strepdb=> select accession, name, molecule from organisms;

 accession |              name               | molecule
-----------+---------------------------------+----------
 AL645882  | Streptomyces coelicolor A3(2)    | chr
 AP005645  | Streptomyces avermitilis MA-4680 | sap1
 BA000030  | Streptomyces avermitilis MA-4680 | chr
 AL589148  | Streptomyces coelicolor A3(2)    | scp1
 AL645771  | Streptomyces coelicolor A3(2)    | scp2
 Ssc       | Streptomyces scabies             | chr
 NC_010572 | Streptomyces griseus             | chr
```

You need the *organisms* to find out accession numbers. Since *vprot* contains information about all the species you need to select information for only the species you want in you query. This is done by specifying an accession in your query.

```
select gene, id, product from vprot where product ~* 'sigma' and
accession = 'AL645882';
```

Or you you want to query on the chromosome as well as the plasmids of *S. coelicolor*

```
select gene, id, product from vprot where product ~* 'sigma' and
accession in (select accession from organisms where name ~* 'coelicolor');
```

3.5 View *vpubs*

This view contains literature references and their associations with different genes. This view is derived from 3 tables (*pubmed*, *journals*, and *ref_link*) and one view (*vprot*).

```
       View "public.vpubs"
  Column    |  Type    | Modifiers
-----------+---------+-----------
 id         | text     |
 gene       | text     |
 accession  | text     |
 serial     | bigint   |
 author     | text     |
 title      | text     |
 journal    | text     |
 pmid       | bigint   |
 pubdate    | integer  |
 pubtype    | text     |
 volume     | text     |
 issue      | text     |
 page       | text     |
 nlmid      | text     |
```

Some column names explained.

- **journal**: This is the full name of the journal.
- **pmid**: This is the pubmed identifier for the article.
- **pubdate**: This is only the year of publication, *not* the complete date.
- **nlmid**: Journals are also given unique identifiers by pubmed.

3.6 View *vtn5*

This view is derived from the tables *features* and *tn5*.

```
  Column    |    Type    | Modifiers
-----------+----------+-----------
 id         | text       |
 accession  | text       |
 serial     | bigint     |
 start_pos  | integer    |
 end_pos    | integer    |
 strand     | smallint   |
 cosmid     | text       |
 cos_pos    | integer    |
 gene       | text       |
 egfp       | boolean    |
```

In this view *start_pos* and *end_pos* columns have identical values.

- **cosmid**: This is the cosmid on which mutagenesis was carried out.
- **start_pos** and **end_pos**: This is the position on the genome where the insertion happened. These two columns have identical values in this table because this is a point location.
- **cos_pos**: Position of the insertion in the cosmid.
- **gene**: If the insertion went into a CDS then this column holds the identifier of the CDS.
- **egfp**: I am not sure about this. Probably indicates whether egfp is in frame or not.

4 Example script

Below is an example Perl script to get information from StrepDB.

```
#!/usr/bin/perl
use strict;
use LWP::UserAgent;

# you might have to uncomment and edit the line below
# if you are behind a proxy.
# $ENV{http_proxy}='http://your.proxy.server:proxyport';
# below is what I have to use.
#$ENV{http_proxy}='http://wwwcache.bbsrc.ac.uk:8080';

# Create a user agent object
my $ua = LWP::UserAgent->new;
# here is our SQL query
my $query="select id, gene, product from vprot where product ~* 'sigma' and accession = 'AL589148'";
# Create a request object
my $request = HTTP::Request->new(POST => 'http://strepdb.streptomyces.org.uk/cgi-bin/strepapi.pl');
# specify the content type
$request->content_type('application/x-www-form-urlencoded');
# load some content
$request->content("query=$query");

# uncomment below to see exactly what is going to the server.
# print("\n", $request->as_string(), "\n");

# Pass request to the user agent and get a response back
my $result = $ua->request($request);
# Check the outcome of the response
if ($result->is_success) {
print $result->content;
}
else {
print $result->status_line, "\n";
}
```

4.1 Example queries for *vpubs*

```
select id, author, pubdate, journal, title from vpubs where gene ~* 'whig';
```

The above query gets all the publications associated with genes which have 'whig' in their *gene* field. The problem with this query is that if three species have genes named *whig* we get all records thrice. This can be dealt with in two ways.

```
select id, author, pubdate, journal, title from vpubs where gene ~* 'whig'
and accession = 'AL645882';
```

The above will ensure that 'whig' of only the chromosome of *S. coelicolor* is considered. Below is another (better?) way of preventing duplicates.

```
select distinct(pmid), author, pubdate, journal, title from vpubs where gene
~* 'whig';
```

The above query simply says that we do not want duplicate *pmid*s. This query will actually look at all *accession*s but suppress duplicate records. Whether you want the *pmid*s or not you have to ask for them in the query. If you do not want them, just ignore them subsequently. It is also important *not* to ask for a column whose value is different in different *accession*s. This will again result in duplicate (or more) *pmid*s being listed. For example in the query below, asking for *id* effectively nullifies the effect of `distinct(pmid)`.

```
select distinct(pmid), id, author, pubdate, journal, title from vpubs where
gene ~* 'whig';
```

4.2 Example queries for *vtn5*

```
select id, start_pos, strand, cosmid, cos_pos, egfp from vtn5;
```

```
select count(*) from vtn5;
```

5 What you get

When you send a request with a SQL query in it, and the query is successful, you are returned a XML document with the information your query retrieved. The root node of this XML document is *recordset* and each record your query retrieved is contained in *record* nodes. So, for example, the query below

```
select id, gene, product from vprot where product ~* 'sigma'
and accession = 'AL589148'
```

returns

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE recordset>

<recordset>
  <record>
    <id>SCP1.116</id>
    <gene>SCP1.116</gene>
    <product>putative ECF-family sigma factor</product>
  </record>
  <record>
    <id>SCP1.151</id>
    <gene>SCP1.151c</gene>
    <product>putative sigma factor</product>
  </record>
</recordset>
```

6 When things don't work

First of all try the example script given in section 4, changing only the line shown below if you need to.

```
$ENV{http_proxy}='http://wwwcache.bbsrc.ac.uk:8080';
```

Unfortunately it is not possible to list all reasons for failure here.

- Make sure your script compiles properly.
- Make sure you have a working internet connection.
- Double check your http proxy setting.

If your are reasonably sure the problem is at my end. Email me. Please include as many details as possible in the mail. Be sure to include the script or program (source) you are using.

7 Caveats

Although significant effort is made to ensure accuracy of information in StrepDB there is no guarantee that there are no errors. Quite a lot of information is computationally generated. *e.g.* references are assigned to genes by a (rather simplistic) computational procedure. Such information will almost certainly have errors in it. You are strongly advised to manually check the suitablity of the information you access from StrepDB for the purpose for which you intend to use it.