How to load a Chado database into BioMart

Authors: Aminah Olivia KELIET[1]
Joëlle Amselem[1]
Sandra DEROZIE[1]
Delphine Steinbach[1]
([1]INRA URGI http://www.urgi.versailles.inra.fr)

Contact: Aminah Olivia KELIET, aminah-olivia.keliet@versailles.inra.fr

Table of Contents

Prerequisites

To load datasets from chado databases, BioMart must be installed in your system.

See instructions in http://www.biomart.org/install-overview.html

BioMart is available in URL http://www.biomart.org.

The BioMart components are available under CVS distribution in two packages :

- **martj :** JAVA API (MartEditor, MartShell, MartExplorer, MartBuilder).
- **biomart-perl :** PERL API (MartView, MartService, DAS Annotation Server).
The installation requires a password: CVSUSER.

To use MartView, MartService and DAS Server, an Apache web server must be installed.

Martj

Martj contains the Java API and whole BioMart applications based on Java. Its directory "bin" contains bash scripts (. sh).
Martj source code is available for download via CVS, but it is necessary to have tool "ant" installed to compile it. "Ant" is available on the site http://ant.apache.org/.
Martj requires at least Java 1.3 (http://java.sun.com/)

Commands to get and install:
- log:
cvs -d :pserver:cvsuser@cvs.sanger.ac.uk:/cvsroot/biomart login
- recover:
cvs -d :pserver:cvsuser@cvs.sanger.ac.uk:/cvsroot/biomart co -r release-0_7 martj

- **Changes :**

In order to fix a problem encountered you need to edit one of the source file:
martj/src/java/org/ensembl/mart/lib/config/DatabaseDatasetConfigUtils.java
Search for the expression "***Remove duplicates before generating template"*** and after the second occurrence delete the expression "return false"
Compilation:
ant jar

Biomart-perl

Commands to get and install:

- log:
cvs –d:pserver:cvsuser@cvs.sanger.ac.uk:/cvsroot/biomart login
- recover:
***cvs –d:pserver:cvsuser@cvs.sanger.ac.uk:/cvsroot/biomart ***
co –r release-0_7 biomart-perl

- Install:
Perl 5.6.0 or more is required (http://www/perl.org/). Biomart-perl depends on a number of Perl modules. To get the list of required and missing modules, launch the command below: (in the directory biomart/biomart-perl)

/usr/local/bin/perl bin/configure.pl -r conf/registryURLPointer.xml

Apache

Apache is available http://httpd.apache.org/ and it is not necessary to configure it, scripts for BioMart do automatically. A version of Apache 1.3 minimum is required. MartView also requires the installation of plug-ins for Apache and a preference for a version higher than 2.0.

To install Apache, you must :

- Create a new directory apache:
  mkdir biomart/apache
- Create a new source directory in biomart:
  mkdir biomart/source
- Retrieve the archive on www.apache.org:
  cd biomart/source
  wget http://www.apache.org/dist/httpd/httpd-2.2.8.tar.gz
- Install it:
  gunzip http-2.2.8.tar.gz
  tar -xvf http-2.2.8.tar
  cd biomart/source/httpd-2.2.8
- Set some environment variables:
  export C-C=cc export CXX=CC export CFLAGS= « -fast –xarch=v9 »
- Run the configuration script :
  ./configure.pl -enable-deflate -prefix=biomart/apache
- Then:
  make
  make test
  make install

The Apache installation is done (biomart/apache/bin) and its configuration.

ModPerl

- Retrieve the archive on http://perl.apache.org/ :
  cd biomart/source
  wget http://perl.apache.org/dist/mod_perl-2.0.3.tar.gz
  gunzip mod_perl-2.0.3.tar.gz
  tar -xvf mod_perl.2.0.3.tar
  cd /home/projects/gpi/biomart/source/mod_perl-2.0.3

- Run the configuration script :

***/usr/local/bin/perl Makefile.pl ***
***PREFIX=/home/projects/gpi/biomart/apache ***
MP_APXS=/home/projects/gpi/biomart/apache/apxs

- Then:

make
make test
make install

BioMart Configuration

1 - Configuring BioMart Perl API.

- Run the configuration script:

cd biomart/biomart-perl
/usr/local/bin/perl bin/configure.pl -r conf/registryURLPointer.xml

- do you want to install in API only mode [y/n] [n]:

y

The final message is “Looks good ... You are done”.

2 - Configuring MartView.

Before using the configuration script MartView must learn some characteristics in settings.conf file in the directory biomart/biomart-perl/conf:

- Apache Binary : biomart/apache/bin/httpd.
- ServerHost : localhost.
- Port : 1111.
- Proxy : none.
- Location : biomart.

This file can set full of other parameters for the web interface (colors, etc ...). The file site_header_biomart.tt in directory biomart/biomart-perl/conf/templates/default allows you to customize the interface (logo, navigation bar, etc ...).

Once everything is customize, the same script is used for configuring MartView:

- Run the configuration script:

cd biomart/biomart-perl
/usr/local/bin/perl bin/configure.pl -r conf/registryURLPointer.xml

- do you want to install in API only mode [y/n] [n] :

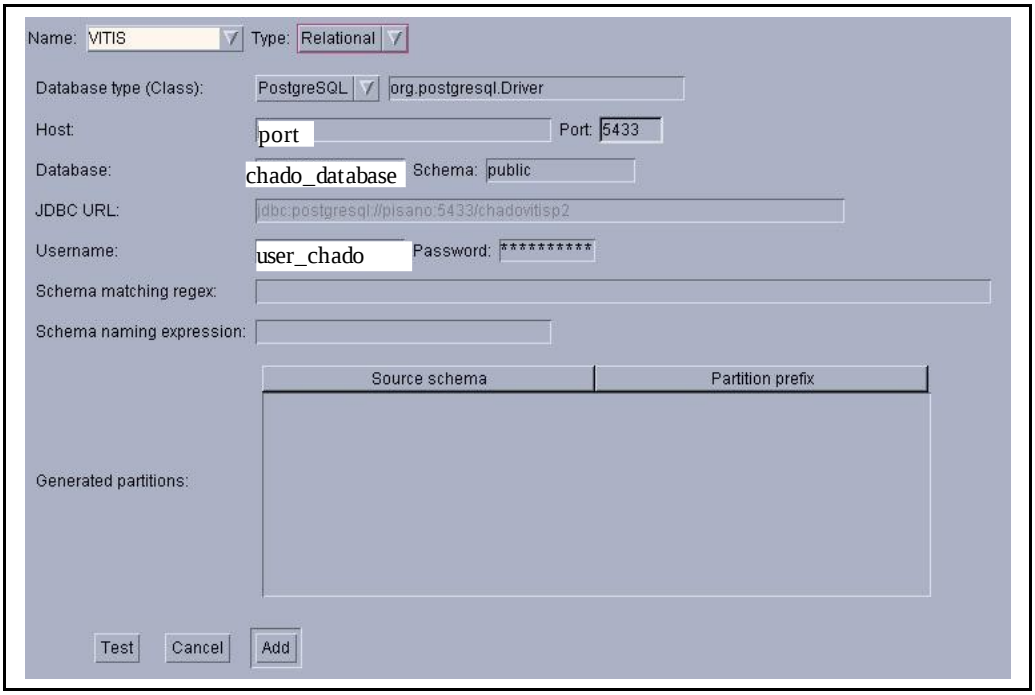
n

Create dataset from a Chado database

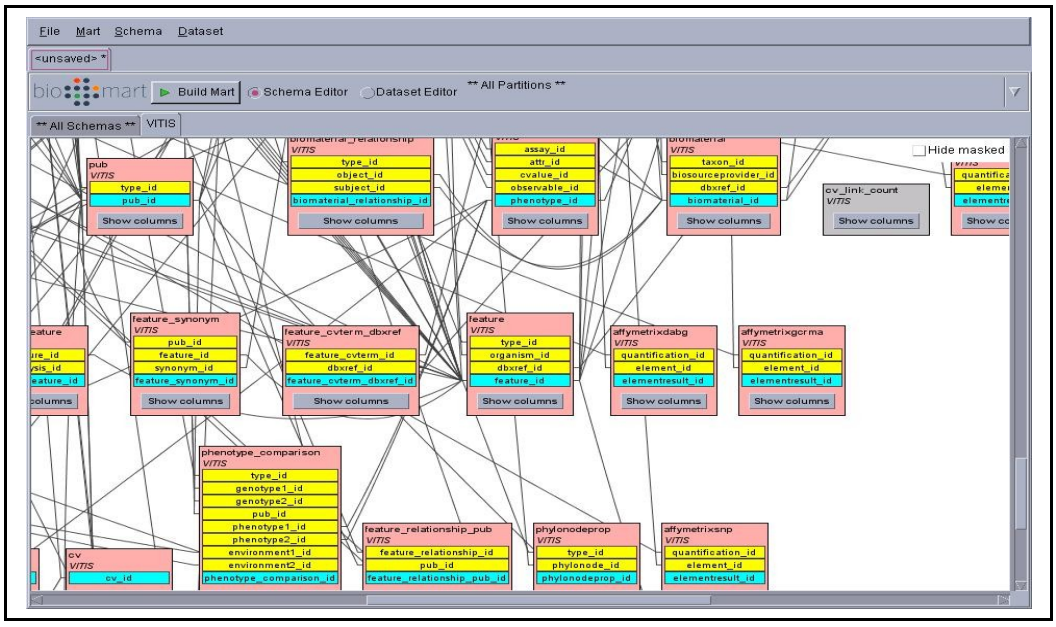
Dataset creation

To create this dataset, a new schema must be created in the chado database. The owner of this schema should have at least right to “select” in chado database and the right to insert data into tables of this new schema.

To create this dataset, the plugin MartBuilder of BioMart was used. Here is the database login window to fill in:



Once the schema database MartBuilder is loaded, just select the central table on which you want to build the dataset. Here, we choose the table "feature" as a central table.



The dataset will be created with the option "Create" menu "Dataset" of MartBuilder, centering the dataset on the table "feature." The dimension tables and the main table are viewable in "Dataset Editor" MartBuilder. This tool create a master table centered on the table feature and twenty five dimension tables.

In this example, some of data will not be recovered, only the following tables will be used:

Analysis	**Feature**
Analysisfeature	**Dbxref**
Cvterm	**Feature_dbxref**
Synonym	**Organism**
Feature_synonym	**Featureloc**

Only the main table feature and dimension tables following will be used for the creation of the dataset:

Analysisfeature	**Feature_dbxref**
Featureloc	**Feature_synonym**

The other dimension tables will be hidden: right click on each table and select the option "Mask."

In order to keep only the tables you need, you should also to make some configuration manual:

-**on the table « feature » :** select "Mask" on the link between “feature” and “dbxref” and on the label "dbxref_id" Tables "feature".

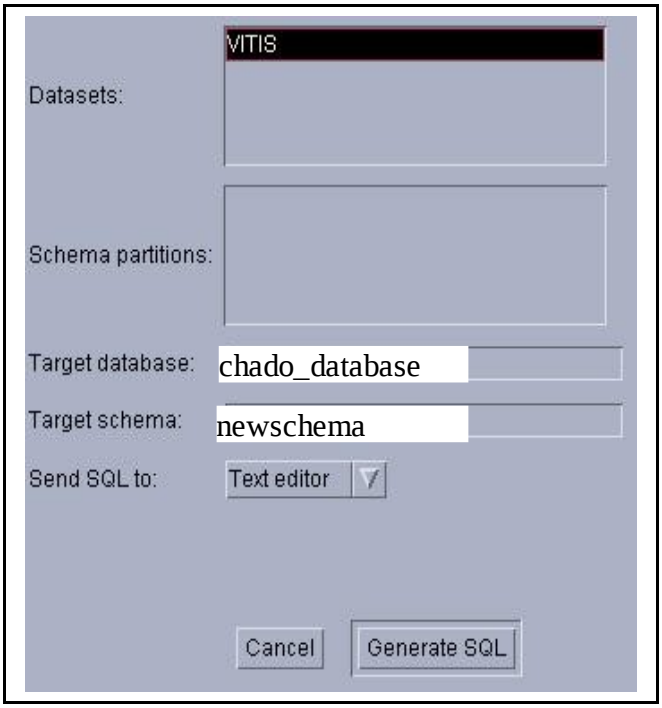
-**on the table « organism » :** select "Mask" on the link between "organism" and "dbxref" and on the label "dbxref_id" table "organism."

-**on the table « feature_synonym » :** select "Mask" on the link between "feature_synonym" and "pub" and the label "pub_id" table "feature_synonym.

-**on the table « synonym » :** select "Mask" on the link between "synonym" and "cvterm" and on the label "cvterm_id" table "synonym".

To check if everything is working as planned, you can use the "explain table ..." on the main table and dimension tables in the "Dataset Editor" MartBuilder. Thus, there is a view of all the merged tables.

To generate the SQL dataset, we use the option "Generate SQL" menu "Mart." Here the window generation of SQL:



With this, SQL statements related to our MartBuilder diagram will be written in a file to save. It is possible to use the software MartRunner. The dataset creation SQL scripts will be executed using the following command:

```
psql -h host -p port -U user -d mydatabase <myfile.sql> myfile.log 2>&1
```

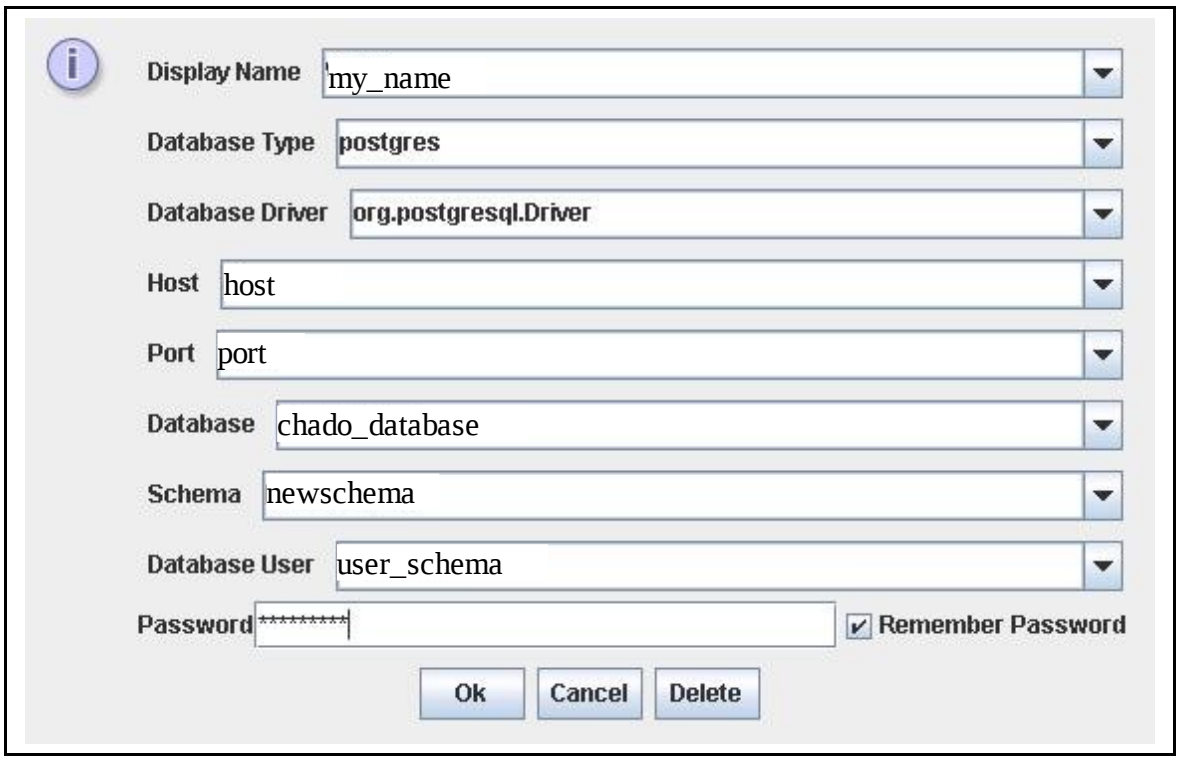
Once the SQL code generated, it is interesting to keep the ". xml" MartBuilder reporting changes on the original schema downloaded.

Indeed, the backup file ". xml" to MartBuilder can work on the layout of the database saved without having to reconnect to it again. This backup contains the whole changes made.

Dataset Configuration

To configure this dataset the plugin MartEditor of BioMart will be used.

Here is the login window that links MartEditor to the database containing the dataset to configure:



Once connected to the database, you can create a basic configuration with "naïve" option (command of menu "File" menu). This will actually create a basic configuration in MartEditor but also insert meta-data tables into the chado database. Once configuration achieved, we have to configure the dataset.

Different pages will be created:

- a "filters" page.
- a "attributes" page.

Filters and attributes can be represented in different ways (See "**Complete BioMart docs in PDF format**" in http://www.biomart.org/install.html). As example use case:

a) Filters page.

This page allows to define on which field the user will be able to run a query. You can choose to group groups filter in categories. We choose to define 3 categories of filters: "analysis", "feature" and "region".

Category	Collection/filtre	Chado DB table.field	Field set up	Comments
Analysis	Analysis_name/analysis_name	Analysis.name	Drop-down list	List of all analysis available in the dataset
	Is_analysis/is_analysis	Feature.is_analysis		Value could be True or False according to the data source, if whether or not linked to an analysis

				stored in analysis table of chado DB
Feature	feature_name/name	Feature.name	Text field /upload file of values	"%" wildcard is accepted. File contains a list of names of several features (features the names must be one per line and the file does not contain blank lines).
	feature_uniquename /uniquename	Feature.uniquename	Text field /upload file of values	"%" wildcard is accepted. File contains a list of names of several features (features the names must be one per line and the file does not contain blank lines).
	type/type	Cvterm.name	Drop-down list	List of types used in feature table (feature.type_id foreign key references cvterm.cvterm_id, eg STS, gene, polypeptide, etc. ...)
Region	Position/fmin	Featureloc.fmin	Text field	positions of start locations of different features
	Position/fmax	Featureloc.fmax	Text field	positions of end locations of different features
	Strand/strand	Featureloc.strand	Drop-down list	List of strand used in featureloc table

b) Page attributes.

The page lets you define attributes on which field the user wishes to obtain the results of its application. We chose to create three categories or groups of filters "feature", "organism", "analysis".

Category	Collection	Attributes	Chado DB table.fields	Comments
Feature	Feature	feature_name	Feature.name	Clickable option with linkoutURL. This allows for a link to the outside, we make a link to a conf Gbrowser of our system information (INRA URGI) (http://urgi.versailles.inra.fr/cgi-bin/gbrowse/grape/?name=%s).
		feature_uniquename	Feature.uniquename	
		type	Cvterm.name	Selected by default in the result with the option "default" validated “true”.
		Phase	Featureloc.phase	
		Strand	Featureloc.strand	
		fmin	Featureloc.fmin	
		fmax	Featureloc.max	
Organism	Feature	abbreviation	Organism.abbreviation	
		common_name	Organism.common_name	
		species	Organism.species	
		genus	Organism.genus	Selected by default in the result with the option "default" validated “true”.
Analysis	Feature	analysis_name	Analysis.name	
		description	Analysis.description	
		program	Analysis.program	

The configuration dataset is well finished.

Operating interface BioMart

To allow the user to query dataset through the BioMart web interface, BioMart MartView tool is used. To connect web interface to the dataset, you have to create an xml file "Registry" containing the connection parameters to the dataset as an example:

```
<?xml version="1.1" encoding="UTF-8"?>
<!DOCTYPE MartRegistry>
<MartRegistry>

<virtualSchema name="default">

<MartDBLocation

name = "my_dataset_name"
displayName = "my dataset name displayed"
databaseType = "oracle"
host = "myhost"
port = "myport"
database = "my_database_name"
schema = "newschema"
user = "user_schema"
password     = "password_schema"
visible      = "1"
default      = ""
includeDatasets = ""
martUser     = ""

/>

<MartDBLocation

name = "my_dataset_name2"
displayName = "my dataset name2 displayed"
databaseType = "postgres"
host = "myhost2"
port = "myport2"
database = "my_database_name2"
schema = "newschema2"
user = "user_schema2"
password     = "password_schema2"
visible      = "1"
default      = ""
includeDatasets = ""
martUser     = ""

/>
```

Here are the steps to configure the web interface to make queries via MartView:

- connection via « .xml » file:

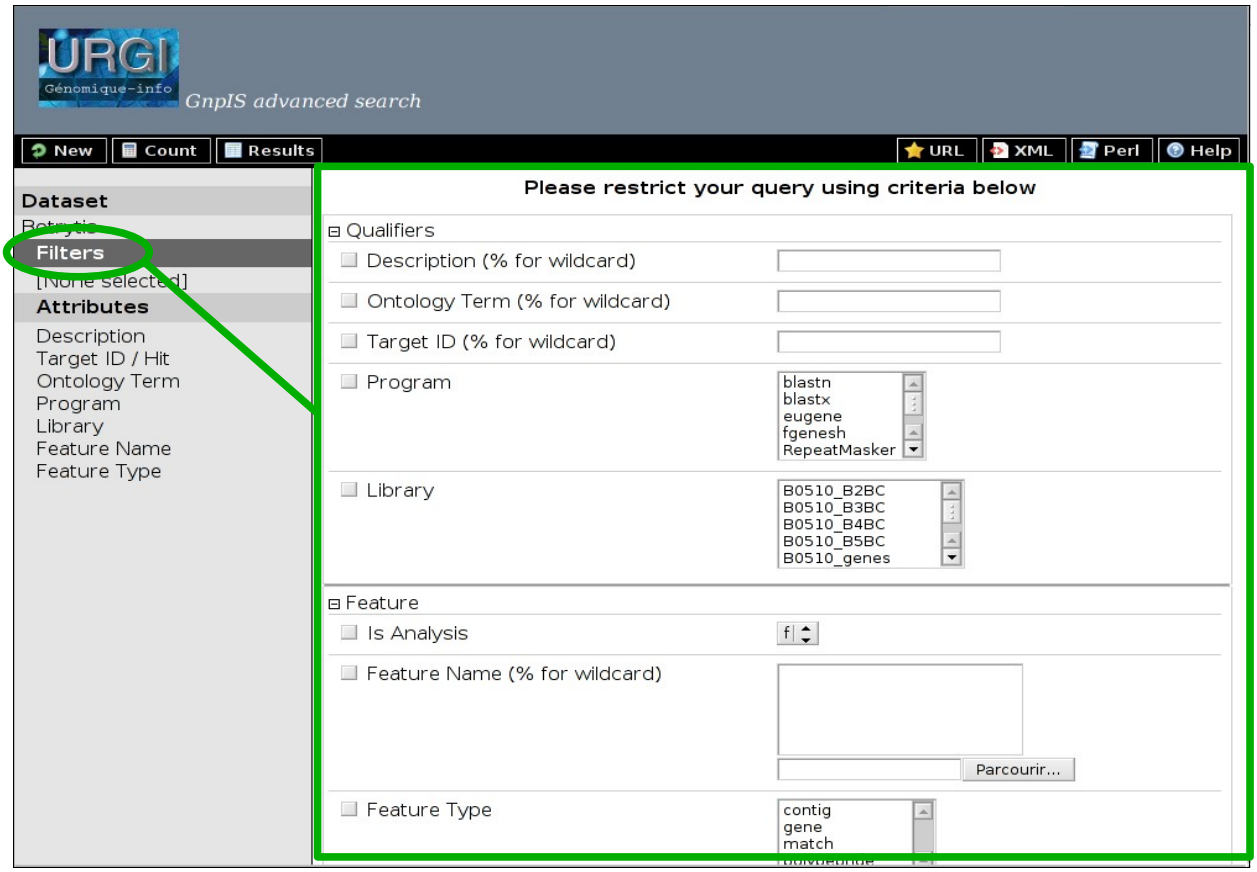
```
export PATH=/usr/local/bin :$PATH
cd /biomart/biomart-perl
perl bin/configure.pl -r conf/myRegistry.xml
```

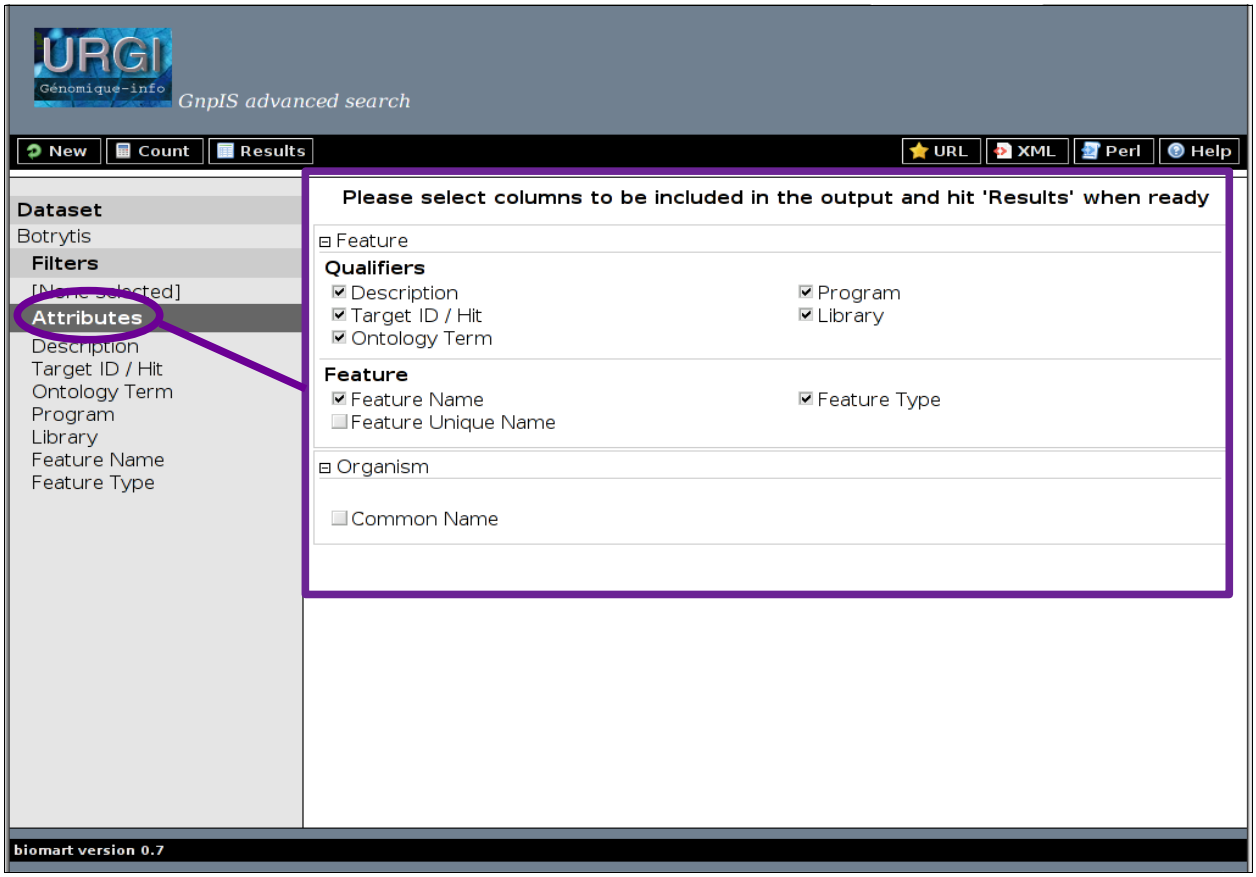
- start Apache :

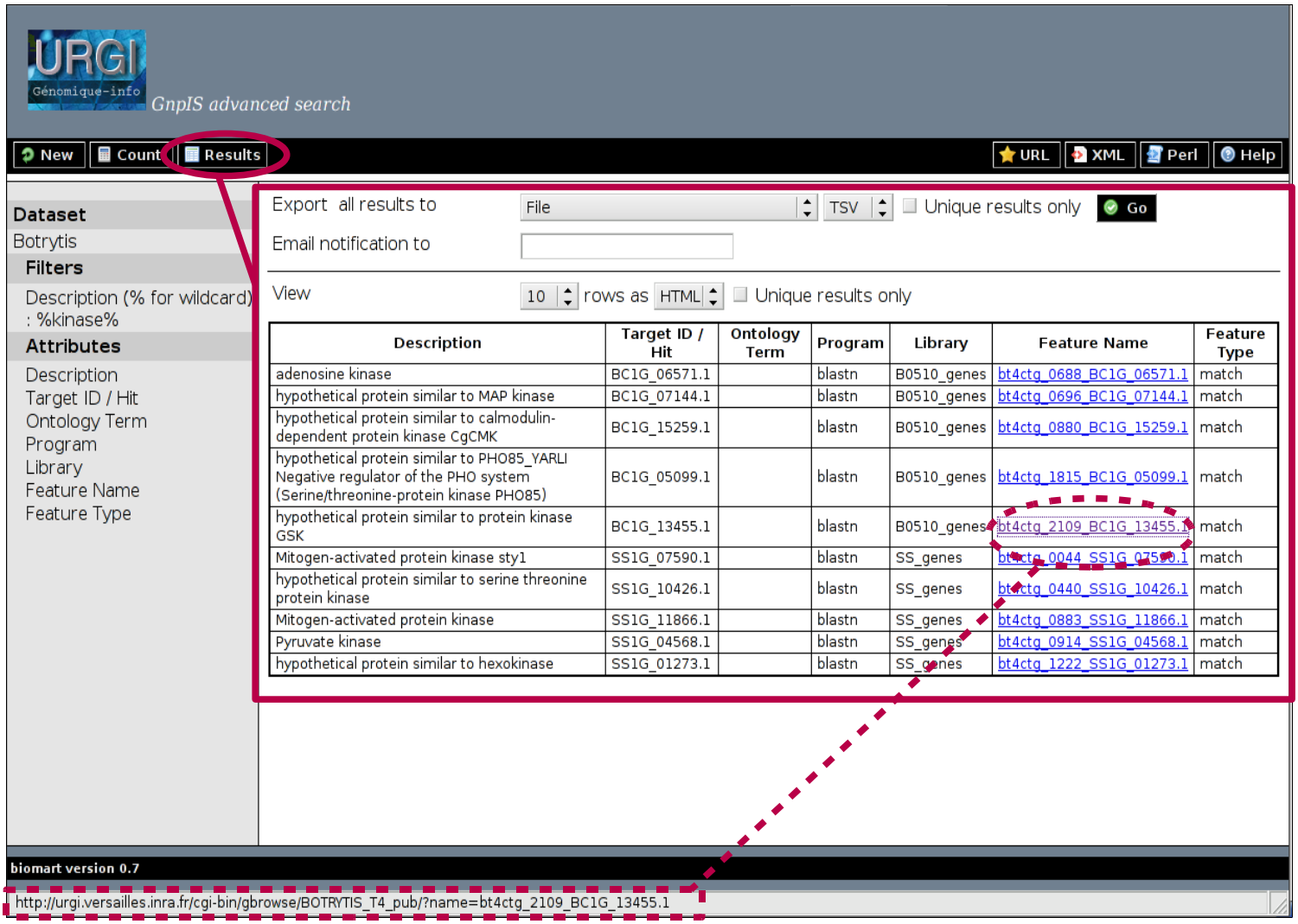
/home/projects/gpi/biomart/apache/bin/httpd –d $PWD –f $PWD/conf/httpd

Once this is done, simply connect to connect to the following address: http://localhost:1111/biomart/martview.

Here is a sample display filters, attributes and results




URGI
Génomique-info
GnpIS advanced search
New
Count
Results
URL
XML
Perl
Help
Dataset
Botrytis
Filters
[None selected]
Attributes
Description
Target ID / Hit
Ontology Term
Program
Library
Feature Name
Feature Type
Please select columns to be included in the output and hit 'Results' when ready
Feature
Qualifiers
Description
Target ID / Hit
Ontology Term
Program
Library
Feature
Feature Name
Feature Unique Name
Feature Type
Organism
Common Name
biomart version 0.7



Description	Target ID / Hit	Ontology Term	Program	Library	Feature Name	Feature Type
adenosine kinase	BC1G_06571.1		blastn	B0510_genes	bt4ctg_0688_BC1G_06571.1	match
hypothetical protein similar to MAP kinase	BC1G_07144.1		blastn	B0510_genes	bt4ctg_0696_BC1G_07144.1	match
hypothetical protein similar to calmodulin-dependent protein kinase CgCMK	BC1G_15259.1		blastn	B0510_genes	bt4ctg_0880_BC1G_15259.1	match
hypothetical protein similar to PHO85_YARLI Negative regulator of the PHO system (Serine/threonine-protein kinase PHO85)	BC1G_05099.1		blastn	B0510_genes	bt4ctg_1815_BC1G_05099.1	match
hypothetical protein similar to protein kinase GSK	BC1G_13455.1		blastn	B0510_genes	bt4ctg_2109_BC1G_13455.1	match
Mitogen-activated protein kinase sty1	SS1G_07590.1		blastn	SS_genes	bt4ctg_0044_SS1G_07590.1	match
hypothetical protein similar to serine threonine protein kinase	SS1G_10426.1		blastn	SS_genes	bt4ctg_0440_SS1G_10426.1	match
Mitogen-activated protein kinase	SS1G_11866.1		blastn	SS_genes	bt4ctg_0883_SS1G_11866.1	match
Pyruvate kinase	SS1G_04568.1		blastn	SS_genes	bt4ctg_0914_SS1G_04568.1	match
hypothetical protein similar to hexokinase	SS1G_01273.1		blastn	SS_genes	bt4ctg_1222_SS1G_01273.1	match

Uses cases: Integration of specific features

Integration of attributes from featureprop table

a) gff3 file

In order to make possible request on feature attributes they should have been inserted in chado DB from column 9 of GFF3 file using the format "tag=value". The "tag=value" are inserted into chado DB featureprop table.

In the example below, "**biological_Process**", "**molecular_Function**" and "**Notes**" will be integrated in BioMart to be filtered on

```
PTR19GenBank_Eugene    polypeptide 162271      165862      .     -     0
     ID=polypeptide|eugene3.00190012_1;biological_Process=apoptosis,defense
response,defense       response       to       pathogen;molecular_Function=ATP
binding;Name=polypeptide|eugene3.00190012_1;Note=gi|15237022|ref|NP_194452.1|
disease resistance protein (NBS-LRR class)%2C putative [Arabidopsis thaliana]
%26gt%3Bgi|46395628|sp|O81825|DR28_ARATH Putative disease resistance protein
At4g27220  %26gt%3Bgi|7486805|pir||T05746  hypothetical  protein  M4I22.30  -
Arabidopsis  thaliana  %26gt%3Bgi|3269283|emb|CAA19716.1|  putative  protein
[Arabidopsis  thaliana]  %26gt%3Bgi|7269575|emb|CAB79577.1|  putative  protein
[Arabidopsis thaliana] (model%25| 68%2C hit%25| 83%2C score| 785%2C %25id| 13)
[Arabidopsis
thaliana];gene=eugene3.00190012;id=573538;interproid=IPR000767,IPR002182;kogid
=KOG4658;product=apoptosis - defense response - defense response to pathogen -
ATP binding;translation=length.1160;Derives_from=mRNA|eugene3.00190012_0
PTR19GenBank_Eugene    exon  162271      164700      .     -     .     ID=exon|
eugene3.00190012_0;gene=eugene3.00190012;Parent=mRNA|eugene3.00190012_0
PTR19GenBank_Eugene    exon  164813      165862      .     -     .     ID=exon|
eugene3.00190012_1;gene=eugene3.00190012;Parent=mRNA|eugene3.00190012_0
```

In Chado DB, these "tags" have been inserted as new terms into cvterm table as follows:

```
cvterm_id     cv_id     name
 ------------  --------  ------------------
 2             1         Note
 26570         5         biological_Process
 26572         5         molecular_Function
```

In chado DB featureprop.type_id Foreign key references cvterm.cvterm_id. Acordingly a supplementary table must be created and will contain the values of the different tag. In the example above "**biological_Process**", "**molecular_Function**" and "**Notes**".

b) SQL code

Thus, the SQL code that should have to be added to the SQL code (myfile.sql) generated by Martbuilder (see ***Creating dataset*** section) could be for the current example:

```
//biological_Process attribute

create  table  newschema.TEMPa  as  select  a.*,b.value  as  value_bioprocess_1078
```

```
from newschema.myname__feature__main as a left join public.featureprop as b on
a.feature_id_1057_key=b.feature_id and b.type_id=26570;
set search_path=newschema,pg_catalog;
create index I_a on newschema.TEMP7(value_bioprocess_1078);
set search_path=newschema,public,pg_catalog;

//molecular_Function attribute

create table newschema.TEMPb as select a.*,b.value as value_molfunct_1078 from
newschema.TEMPa as a left join public.featureprop as b on
a.feature_id_1057_key=b.feature_id and b.type_id=26572;
set search_path=newschema,pg_catalog;
drop table newschema.TEMPa;
set search_path=newschema,pg_catalog;
create index I_b on newschema.TEMP7d(value_molfunct_1078);
set search_path=newschema,public,pg_catalog;

//Note attribute

create table newschema.TEMPc as select a.*,b.value as value_note_1078 from
newschema.TEMPb as a left join public.featureprop as b on
a.feature_id_1057_key=b.feature_id and b.type_id=2;
set search_path=newschema,pg_catalog;
drop table newschema.TEMPb;
set search_path=newschema,pg_catalog;
create index I_c on newschema.TEMP7d(value_note_1078);
set search_path=newschema,public,pg_catalog;

//creating of __featureprop__dm table

set search_path=poplarmart,poplarmart,pg_catalog;
alter table poplarmart.TEMP7c rename to myname__featureprop__dm;
set search_path=newschema,pg_catalog;
create index I_d on newschema.myname__featureprop__dm(feature_id_1057_key);
set search_path=newschema,newschema,pg_catalog;
```

After running the **psql** command to execute the sql (myfile.sql), the datasets from the Chado DB will be created with attribute values to be integrated into Biomart. These data are contained in the "myname__featureprop__dm" table.
To add these new attributes in configuration, see ***Configuration of dataset*** section.

Integration of Target

a) gff3 file

The aim here is to allow the user to make a request on a Target (ie attributes inserted in chado DB relative to the Target). The Target corresponds to Hit in a comparison analysis. So the ID, match start, match end, description or every other attribute inserted in the database relative to the Target could be request under BioMart.
The most common way to insert data relative to a Target is to insert it through GFF3 as "Target" attributes in match and match_part types as Target_ID+start+end.

Examples of GFF3 required to insert a blast analysis result are showed below:

GFF3 corresponding to the Reference feature inserted once

```
bt4ctg_0002       Genoscope      contig     1       20000      +       .
.       ID= bt4ctg_0002;Name= bt4ctg_0002
```

GFF3 corresponding to the Target feature inserted once

```
.       Bot_allest      EST     .       .       .       .       .
ID=PD0ACA5YN07FM1;description=BT4 mycelium - pH stress library
```

GFF3 corresponding to the match between reference feature and the target feature

```
bt4ctg_0002     blastn_Bot_allest       match_set       6244    7015    0.0
+       .  ID=blastn_BT4_PD0ACA_bt4ctg_0002_PD0ACA5YN07FM1_m1;
Target=PD0ACA5YN07FM1+1+725;Name=bt4ctg_0002_PD0ACA5YN07FM1;target_pcover=100.
00;target_pident=100.00;target_length=725;lib=BT4_PD0ACA;program=blastn
bt4ctg_0002     blastn_Bot_allest       match_part      6483    7015    0.0
+       .
ID=blastn_BT4_PD0ACA_bt4ctg_0002_PD0ACA5YN07FM1_m1_mp1;Target=PD0ACA5YN07FM1+1
93+725;Parent=blastn_BT4_PD0ACA_bt4ctg_0002_PD0ACA5YN07FM1_m1
bt4ctg_0002     blastn_Bot_allest       match_part      6244    6435    1e-104
+     .     ID=blastn_BT4_PD0ACA_bt4ctg_0002_PD0ACA5YN07FM1_m1_mp2;
Target=PD0ACA5YN07FM1+1+192;Parent=blastn_BT4_PD0ACA_bt4ctg_0002_PD0ACA5YN07FM
1_m0001
```

"Reference" feature and "Target" feature are inserted into the chadoDB "feature" table. Information relative to the match location on Target feature and on Reference feature is stored in “featureloc” table.
In the example above :
bt4ctg_0002 is the reference feature
PD0ACA5YN07FM1 Target feature:
blastn_BT4_PD0ACA_bt4ctg_0002_PD0ACA5YN07FM1_m1 is the ID of the match feature between reference and target feature..

To allow the addition in BioMart query builder of a filter on Target IDs, or other attributes relative to Target we need to get all the features for which they are target in order to add a link on the result displayable in GBrowse.

Supposing that we would like to search all the region mapped by the Target PD0ACA5YN07FM1, search for this Target name in the table feature.

```
select   feature_id,   uniquename,   name   from   feature   where   name   =
'PD0ACA5YN07FM1';
 feature_id |   uniquename   |      name
------------+----------------+----------------
      61564 | PD0ACA5YN07FM1 | PD0ACA5YN07FM1
```

Search now for the features in which PD0ACA5YN07FM1 is Target (rank for Reference features are 0 and rank for Target are 1.

```
select  feature_id,  srcfeature_id,  fmin,  fmax,  rank  from  featureloc  where
feature_id in
    (select feature_id from featureloc where srcfeature_id = 61564);
feature_id | srcfeature_id | fmin | fmax | rank
------------+---------------+------+------+------
     257075 |         61564 |  192 |  725 |    1
```

```
     257075 |          121 | 6482 | 7015 |    0
     257074 |        61564 |    0 |  192 |    1
     257074 |          121 | 6243 | 6435 |    0
     257073 |        61564 |    0 |  725 |    1
     257073 |          121 | 6243 | 7015 |    0
     532911 |        61564 |  192 |  725 |    1
     532911 |          121 | 6482 | 7015 |    0
     532910 |        61564 |    0 |  192 |    1
     532910 |          121 | 6243 | 6435 |    0
     532909 |        61564 |    0 |  725 |    1
     532909 |          121 | 6243 | 7015 |    0
```

Search now for all the uniquename of feature_id of for which PD0ACA5YN07FM1 is Target

```
select feature_id, uniquename from feature where feature_id in
       (select feature_id from featureloc where feature_id in
             (select feature_id from featureloc where srcfeature_id = 61564));
 feature_id |                                uniquename
------------
+-------------------------------------------------------------------
     257073 | blastn_BT4_PD0ACA_bt4ctg_0002_PD0ACA5YN07FM1_m1
     257075 | blastn_BT4_PD0ACA_bt4ctg_0002_PD0ACA5YN07FM1_m1_mp1
     257074 | blastn_BT4_PD0ACA_bt4ctg_0002_PD0ACA5YN07FM1_m1_mp2
     532909 | sim4_BT4_PD0ACA_bt4ctg_0002_PD0ACA5YN07FM1_m1
     532911 | sim4_BT4_PD0ACA_bt4ctg_0002_PD0ACA5YN07FM1_m1_mp2
     532910 | sim4_BT4_PD0ACA_bt4ctg_0002_PD0ACA5YN07FM1_m1_mp1
```

Finally we get the features (here matches from blastn and sim4 analysis) on which we will make hypertext link to display this region in GBrowse). Thus we have to add a table to integrate BioMart Targets in addition of those generated by MarBuilder. This new table will contain contain values for each Target features.

b) SQL code

Thus, the SQL code will have to be added to the SQL code (myfile.sql) provided by the MartBuilder (see ***Creating dataset*** section) could be:

```
create table newschema.table1 as
select feature_id_1057_key as feature_id, name_1057 as name_feature_1057,
type_id_1057
from newschema.myname__feature__main

--Search features that have multiple sources (b.rank=1)

create table newschema.table2c as select b.feature_id as
feature_id_1057_key,a.name_feature_1057,a.feature_id,b.srcfeature_id
from newschema.table1 as a left join public.featureloc as b on
b.feature_id=a.feature_id and b.rank=1;
drop table newschema.table1;

--Search for names of sources (name_target) of features that have multiple
sources

create table newschema.table3 as select a.feature_id_1057_key,b.name as
name_target_1057 from newschema.table2c as a left join public.feature as b
on a.srcfeature_id=b.feature_id;
```

```
drop table newschema.table2c;

create table newschema.table3d as select distinct b.feature_id_1057_key,
a.name_target_1057 from newschema.myname__feature__main as b left join
newschema.table3 as a on a.feature_id_1057_key=b.feature_id_1057_key;
drop table newschema.table3;

--creating __target__dm table

set search_path=newschema,newschema,pg_catalog;
alter table newschema.table3d rename to myname__target__dm;
set search_path=newschema,pg_catalog;
create index I_t on newschema.myname__target__dm(feature_id_1057_key);
set search_path=newschema,newschema,pg_catalog;
```

After running the **psql** command to execute the sql (myfile.sql), datasets from the Chado DB will be created, with values of targets that must be integrated into BioMart. These data are contained in the "myname__target__dm" table.
For dataset configuration , see ***Configuration of dataset*** section .

Integration of Ontology Terms

a) gff3 file

To make request on attributes of type "Ontology_term" using in BioMart query Builder. We suppose that this tag was present in GFF3 file used for insertion into Chado database.
An exemple is showed below:

```
PTR03eugene      gene  12312 12445 .     +     .     ID=gene_Oli|
eugene;Ontology_term="GO:0046703"
```

Given that chado DB Ontologies Terms are inserted into the "cvterm" table, the table "cvterm" is joined to the "feature" table by "feature_cvterm" table;

cvterm.cvterm_id=feature_cvterm.cvterm_id

and

feature_cvterm.feature_id=feature.feature_id

Thus to integrate Ontologies Term in Biomart, a table should be created and added to those provided by MarBuilder . This table Ontology terms values for each feature.

b) SQL Code

Thus, the sql code which must be added to the sql code (myfile.sql) that generates Martbuilder (see ***Creating dataset*** section) can be:

```
create table newschema.TEMP0001 as select a.feature_id,b.name from cvterm b,
feature_cvterm a where b.cvterm_id=a.cvterm_id;
set search_path=newschema,pg_catalog;
create table newschema.TEMP000 as select a.*, b.name as ontologie_name_1057
from newschema.TEMP0 as a left join newschema.TEMP0001 as b on
a.feature_id_1057_key=b.feature_id;
set search_path=newschema,pg_catalog;
```

```
drop table newschema.TEMP0;
drop table newschema.TEMP0001;
set search_path=newschema,pg_catalog;
create index I_00 on newschema.TEMP000(cvname_1057);
```

This code must be integrated into the sql code of __feature__main table creation, as follows:

```
set search_path=newschema,public,pg_catalog;
create table newschema.TEMP0 as select a.residues as
residues_1057,a.is_analysis as is_analysis_1057,a.organism_id as
organism_id_1057,a.is_obsolete as is_obsolete_1057,a.uniquename as
uniquename_1057,a.type_id as type_id_1057,a.feature_id as
feature_id_1057_key,a.name as name_1057 from public.feature as a;
set search_path=newschema,pg_catalog;
create index I_0 on newschema.TEMP0(type_id_1057);
set search_path=newschema,public,pg_catalog;

                              HERE


set search_path=newschema,public,pg_catalog;
create table newschema.TEMP1 as select a.*,b.dbxref_id as
dbxref_id_1037,b.definition as definition_1037,b.name as
name_1037,b.is_obsolete as is_obsolete_1037,b.cv_id as
cv_id_1037,b.is_relationshiptype as is_relationshiptype_1037 from
newschema.TEMP000 as a left join public.cvterm as b on
a.type_id_1057=b.cvterm_id;
set search_path=newschema,pg_catalog;
drop table newschema.TEMP000;
set search_path=newschema,pg_catalog;
create index I_1 on newschema.TEMP1(organism_id_1057);
set search_path=newschema,public,pg_catalog;
create table newschema.TEMP2 as select a.*,b.species as
species_1084,b.common_name as common_name_1084,b.genus as genus_1084,b.comment
as comment_1084,b.abbreviation as abbreviation_1084 from newschema.TEMP1 as a
left join public.organism as b on a.organism_id_1057=b.organism_id;
set search_path=newschema,pg_catalog;
drop table newschema.TEMP1;
set search_path=newschema,newschema,pg_catalog;
alter table newschema.TEMP2 rename to myname__feature__main;
set search_path=newschema,pg_catalog;
create index I_2 on newschema.myname__feature__main(feature_id_1057_key);
```

After running the **psql** command to execute the SQL code (myfile.sql), the dataset from Chado DB will be created, with values of Ontology Term to be integrated into BioMart. These data are contained in the “myname__feature__main” table as name "**ontologie_name_1057**”.

For this dataset configuration, see ***Configuration of dataset*** section.