Implementation of VO queries

Matthew Whiting

Contents

1 Summary

This document summarises how the VO-compliant query form is to be set up, what queries we expect, and what modifications need to be made from where the archive is at present (June 2007). This will be continually updated as this part of the project progresses.

2 Expected queries

Here is a list of expected queries that the archive should be able to deal with, along with comments on each:

- All observations of a given pulsar.
  This is essentially how the archive works at the moment.
- All observations of a given sky position, within a search radius – this is essentially a conesearch.
  This also exists on the archive at the moment, but doesn't seem to work. An example: a search for J0835-4510 returns a large number of files, but a search at its RA & Dec gives nothing (i.e. "No pulsars found within (08:35:20.700, -45:10:35.700) with radius 2").
- Accept standard conesearch queries from a VO service.
  Not possible at present.
- All observations within given time range, perhaps combined with above.
  A time-range is catered for at present – and it seems to work fine.
- Be able to refine queries based on pulsar parameters.
  i.e. search for all pulsars with period in a given range, or with DM greater than some value... This is not catered for at present.

This is mostly a wish-list. The current project (as at June 2007) will involve enabling conesearches to be made on survey data, as well as the queries that are currently possible with the on-line archive. Conesearches on timing data, particularly with concurrent searches on pulsar parameters is probably beyond the scope of what we can do.

3 The mySQL database

We note here some important points for the mySQL database. It is based on `herschel`, with two accounts: **pulsar** and **psrdba** – the latter is the administrative account and should be used for adding/modifying tables and so on. The first allows searching of tables and other similar queries only.

The full set of php, sql and related files that Albert has compiled reside on the machine `herschel`, in `herschel:/var/www/vhosts/pulsararchive.atnf.csiro.au/htdocs` and sub-directories. This includes the perl script `perl/find_archives.pl`, which does the trawling of the disks to pick up new data files.

3.1 Problems

- When you are on herschel, you cannot see the `/psr` disk, which means all the psr programs are unavailable. Is there any reason for this? It is particularly a problem with the position-related query on the web archive, as this uses `vap` to search for pulsars around a given position.
- The version of mySQL could probably do with an update. It is only 4.0, and it would be great to have 5.0 or similar so that we could create functions easily – I'm thinking functions like an angular separation calculator, as this would make the cone-search queries much easier to write.

4 Existing data: The "observations" table

The list of the columns in the existing `observations` mySQL table is given in the table in Appendix A. There is a second table, `observations2` that has different data in it – it is not clear exactly what this data is or where it comes from (there is no reference to it in any of the files in the `htdocs` directory on `herschel`. Its setup (which is slightly different to `observations`) is also shown in Appendix A.

4.1 Missing and incomplete data

There are several columns that have no information – that is, every row in the table is NULL. They are: date, ut, gl, gb, tsamp, nbeam.

4.2 Positional information

Note that the sky positions for the files in the `observations` table are broken up into hours/degrees, minutes and seconds columns. This presents a problem, as the declination information does not properly record positions like -00:01:23. This is because you cannot distinguish between -0 and $+0$ in the decj_degree entry.

The positions should either be stored as decimal degrees (best option), or, if the hours/degrees, minutes and seconds must be stored separately, also have a decj_sign column that indicates the sign of the declination (equal to $+1$ or -1). The positions could also be stored as a string (i.e. "18:23:45.3"), although a numerical value will also be needed for angular separation calculations.

An example of how we can create new columns to store the decimal degree values is the following bit of SQL code:

```
SELECT pulsar_name,filename,data_type,decj_degree,decj_minute,decj_second,
        (raj_hour+raj_minute/60.+raj_second/3600.)*15 as raj,
        CASE WHEN pulsar_name LIKE "%-%"
             THEN -1.*(abs(decj_degree)+decj_minute/60.+decj_second/3600.)
             ELSE abs(decj_degree)+decj_minute/60.+decj_second/3600.
             END as decj
FROM observations
WHERE abs(decj_degree)<1
      AND (decj_minute>0 or decj_second>0)
ORDER BY pulsar_name;
```

Currently the positions are stored incorrectly even in the ATNF Pulsar Database[1]. A search "within circular boundary" with radius of 1 degree from 16:07:00, -00:30:00 does not return J1607−0032, which appears to be 1.05 degrees away. This is because the DECJ is returned as +00:32:40.83 (i.e. the sign of the declination is not preserved). This seems to be a wider problem, originating with the `psrcat` program.

A new table can be created very simply that has two new columns with the decimal values of RA and DEC in it. We can also put values:

```
-- version of mysql is <4.1 so cannot use CREATE ... LIKE ...
CREATE TEMPORARY TABLE newobs
  SELECT *,
      (raj_hour+raj_minute/60.+raj_second/3600.)*15 as raj,
      CASE WHEN pulsar_name LIKE "%-%"
               THEN -1.*(abs(decj_degree)+decj_minute/60.+decj_second/3600.)
           WHEN (decj_degree<0.)
               THEN -1.*(abs(decj_degree)+decj_minute/60.+decj_second/3600.)
           ELSE abs(decj_degree)+decj_minute/60.+decj_second/3600.
      END AS decj
  FROM observations;
```

Note the two clauses that set the declination to be negative. The first is designed to deal with sources that have $-1^\circ < \delta < 0^\circ$, and looks at the name to see if it has a minus sign. The second clause deals with sources

[1]See http://www.atnf.csiro.au/research/pulsar/psrcat/

that don't have an "IAU-format" name (such as "CAL"). This will only catch them if they have $\delta < -1°$, but fortunately all the sources with $-1° < \delta < 0°$ are suitably named to be caught by the first clause.

The only remaining problem sources as far as positions go are those with no positional information in the table. There are 246 rows that satisfy this criterion – how are we going to deal with these? If they have a pulsar name, then we can access the positions somehow and manually put them in. However that may be a somewhat labourious process...

While adding in the decimal RA & Dec, we could also add in the proper values for the galactic coordinates. These have not been put in the database for most/all(?) of the existing data, and so we could do it all at once. The code is shown in `sql/create_new_obs_table.sql`: it first creates a temp copy of `observations`, with RA and DEC but without GL and GB. It then creates a new table with the following code, calculating the values of GL and GB:

```
SET @ngpra=192.859508, @ngpdec=27.128336, @ascnode=32.932; -- base coords for galactic coord transf

CREATE TEMPORARY TABLE newobs
   SELECT *,
        case when degrees( atan2( ( sin(radians(decjd))*cos(radians(@ngpdec)) - cos(radians(decjd))
     then degrees( atan2( ( sin(radians(decjd))*cos(radians(@ngpdec)) - cos(radians(decjd))*sin(rad
     else degrees( atan2( ( sin(radians(decjd))*cos(radians(@ngpdec)) - cos(radians(decjd))*sin(rad
(degrees( asin( cos(radians(decjd))*cos(radians(@ngpdec))*cos(radians(rajd-@ngpra)) + sin(radians(d
    FROM temptab;
```

5 The new data: 70cm Survey

5.1 Location of data

The data is going to be stored in the $S70V directory, once it is converted to PSRFITS format from the PKS format files in $S70TAPES. There is some data there now, most of which is not up to date with all the metadata (it was converted earlier in the year, with older versions of the PVOCONVERT code and the PSRFITS definition file), and some that I have converted in the past week.

5.2 Incorporating the data into the database

It is relatively straightforward to load the data into the database, either directly using SQL commands or via the PHP interface. Here's an example of how it may be done (taken from `$PVO/sql/create_table_s70.sql`):

```
-- create the table with the following fields
CREATE TEMPORARY TABLE s70 (
file_id SMALLINT UNSIGNED AUTO_INCREMENT,
filename VARCHAR(30),
fileloc VARCHAR(100),
hdrver VARCHAR(10),
survey VARCHAR(10),
src_name VARCHAR(20),
        telescope VARCHAR(16),
        data_type VARCHAR(16),
raj VARCHAR(15),
decj VARCHAR(15),
rajd DOUBLE,
decjd DOUBLE,
gl DECIMAL(9,4),
gb DECIMAL(9,4),
bmaj FLOAT,
bmin FLOAT,
bpa FLOAT,
date_obs VARCHAR(30),
mjd DOUBLE,
obsfreq DOUBLE,
chanbw DOUBLE,
nchan INT,
scanlen DOUBLE,
tsamp DOUBLE,
        nbits INT,
PRIMARY KEY (file_id)
);

-- now load the data from the file in my PulsarVO directory.
-- It is a csv file, and the strings in it are enclosed by ".
-- We don't want the quote marks in the table, so the
-- "OPTIONALLY ENCLOSED BY" statement will remove them.
LOAD DATA LOCAL
INFILE "/DATA/SITAR_1/whi550/VO/PulsarVO/s70v_summary.csv"
INTO TABLE s70
FIELDS TERMINATED BY ","
OPTIONALLY ENCLOSED BY "\""
IGNORE 1 LINES  -- ignore the first line in the csv file (contains the headers)
(filename,fileloc,hdrver,survey,src_name,raj,decj,rajd,decjd,gl,gb,
bmaj,bmin,bpa,date_obs,mjd,obsfreq,chanbw,nchan,scanlen,tsamp,nbits);
```

The exact columns are still to be fully determined, but the SQL makeup will be as given in Appendix B.

6 VO conesearch queries

6.1 How it can be done in SQL code

Here is an example of how you can query the database, once the 70cm data is in there, by searching for angular separation:

```
-- Performing a conesearch around the position
--  RA=0.=00:00:00 and DEC=-43.=-43:00:00 for the S70 data.
-- This should pick up three files from the new batch added on June 26.
-- The old query of RA=0 and DEC=-49 should pick up four files.
SET @qra=0., @qdec=-49.;
SELECT filename, raj, decj, 0.5*(bmaj+bmin) AS beamsize, obsfreq, scanlen,
       (degrees( acos( cos(radians(rajd-@qra))*
                       cos(radians(decjd))*cos(radians(@qdec))
                      +sin(radians(decjd))*sin(radians(@qdec)) ) )) AS angsep
FROM s70
WHERE (degrees( acos( cos(radians(rajd-@qra))*
                      cos(radians(decjd))*cos(radians(@qdec))
                     +sin(radians(decjd))*sin(radians(@qdec)) ) )
        - 0.25*(bmaj+bmin) ) <  1.
ORDER BY angsep;
```

6.2 Producing a VOTable

The above SQL structure can be readily incorporated into a .php file and used to query the database via a dynamic webpage. This .php file can also produce an XML file rather than the usual HTML format. The way to do this is to have the following expression at the start of the php file:

```
header("Content-type: text/xml");
```

to make sure the resulting file is read as XML.

From there, it is straightforward to put in the necessary VOTable fields and fill in the data that results from the query. This is done in the file `conequery-vo.php`.

6.3 Provision of data – what's in the VOTable?

The VOTable that we provide is simply a table of metadata. By default all metadata that is in the SQL table is provided, with the exception of the file's disk location. We do provide the filename, but that is all – there is no mechanism to directly obtain the data from the cone-search query.

The reasoning here is that a user might first do a VO cone-search, and then, if a result of interest came up, they would then go to the web archive and repeat the search. The web archive's results page would then provide means for downloading the data, in much the same way as it is set up at present (thanks to Albert).

6.4 Registering the VO cone search service

For the conesearch service to be widely accessible, it should be registered with a VO registry. There are a number of such registries, which typically have forms that allow registration of new services. Two sites that provide links to registries are:

1. NVO at http://www.us-vo.org/pubs/files/PublishHowTo.html, linking to three different NVO registries.

2. Euro-VO, at http://www.euro-vo.org/pub/tc/registries.html, which also links to three different registries.

It is unclear whether all these registries are linked together, in such a way that registering at one provides access through all. This is the aim of VO registries, but it is not apparent whether this is reality yet. This needs to be sorted out.

The process of registering a service involves providing the registry with metadata about the service. The full list of Resource Metadata can be found in the IVOA document "Resource Metadata for the Virtual Observatory", at http://www.ivoa.net/Documents/latest/RM.html.

7 The web query form

7.1 Structure

This section will layout the rough format of the query form that acts as the front-end to the archive. The following options should exist:

- Position query. RA & Dec should be the default. May be useful to offer the possibility of a query in Galactic coordinates. Will need to be able to switch between these.

  Also need a search radius box. Default units in arcmin? degrees? Allow this to be selectable?

- Pulsar name. This should accept a pulsar name, be able to deal with a leading letter (e.g. J1234−5678 and 1234−5678 should act the same). This functionality exists, although check that it uses SQL directly and not vap or similar.

- Time of observation. This should be a range, default values being the range of observation dates in the database (although maybe make final date today's date?).

  It may be good to make this field selectable as well, so that one can query with MJD rather than date.

- Refinement of returned parameters. Since there are different tables with different fields, the way to deal with this might be to make a master list of fields to select from. The queries can then only refer to the relevant ones for their table and set the others to NULL or similar (can we use something like *left outer join*?).

  Perhaps a format similar to that used by Vizier[2], where it provides a default list of parameters but allows selection of more and querying on them as well.

  The way it is currently set up does not allow querying on these parameters. This is beyond the scope of the current project, although may be worthwhile implementing in the future. Something similar is enabled with the pulsar database (i.e. querying on *pulsar* parameters such as period, DM, period-derivative), although there is no data access involved. The difference here is that one would be able to query on the parameters related to the observations (such as `tsamp, nchan`).

- The existing webpage has the possibility of restricting the query by specifying what type of data to look for (CPSR2, DFB, WBC). This should be extended to include FB_1BIT (as for the 70cm data) and MFB (for multibeam-filterbank – uncertain exactly what to call this at this stage). These need to be included in the database ingest, and be extracted from the database in making the query page.

  The selection of data type could allow more than one selection – this will change how the selection is done in the sql query.

7.2 Authentication

Currently, the authentication of users is done via a `users` table in the mySQL database. A better way would be to do something similar to that used by ATOA[3] This accesses the OPAL database to authenticate users, and also allows proprietary access to data. This is achieved by associating a Project ID with each file, and having a list of accredited observers associated with each Project ID. If the user is in the list of observers for the ID listed with the file in question, the user is allowed to access the file. Otherwise, the file is marked as unavailable.

The way the OPAL database is accessed can be hidden from the main archive web page, and be wrapped up in a single php function call. This part should be straightforward.

The determination of proprietary access will require a bit of work, as the Project ID is currently **not** stored in the mySQL database. This will need to be incorporated and stored as part of the upkeep of the database.

[2]http://vizier.u-strasbg.fr/cgi-bin/VizieR – see an example catalogue at http://vizier.u-strasbg.fr/viz-bin/VizieR?-source=VIII/34 (select "catalog").

[3]Australia Telescope Online Archive, http://atoa.atnf.csiro.au.

A The "observations" table setup

Field	Type	Null	Key	Default	Extra
pulsar_name	varchar(16)				
filename	varchar(32)		PRI		
raj_hour	int(2)	YES		0	
raj_minute	int(2)	YES		0	
raj_second	decimal(12,10)	YES		0.0000000000	
decj_degree	int(2)	YES		0	
decj_minute	int(2)	YES		0	
decj_second	decimal(12,10)	YES		0.0000000000	
dm	decimal(16,10)	YES		NULL	
period	decimal(32,24)	YES		NULL	
bw	decimal(16,6)	YES		NULL	
cnfg	varchar(128)	YES		NULL	
data_loc	varchar(128)		PRI		
data_type	varchar(16)	YES		NULL	
freq	decimal(32,16)	YES		NULL	
inst	varchar(32)	YES		NULL	
MJD	decimal(32,26)	YES		NULL	
npol	int(6)	YES		NULL	
nchan	int(6)	YES		NULL	
nbin	int(6)	YES		NULL	
nsub	int(6)	YES		NULL	
rcvr	varchar(16)	YES		NULL	
site	varchar(8)	YES		NULL	
file_size_bytes	int(32)	YES		NULL	
length	decimal(32,20)	YES		NULL	
obsrvr	varchar(32)	YES		NULL	
date	varchar(16)	YES		NULL	
ut	varchar(16)	YES		NULL	
gl	float	YES		NULL	
gb	float	YES		NULL	
tsamp	float	YES		NULL	
nbeam	tinyint(3) unsigned	YES		NULL	

Table 1: Fields for the `observations` table in the mySQL database, as it stands at present.

Field	Type	Null	Key	Default	Extra
pulsar_name	varchar(16)				
filename	varchar(32)				
raj_hour	int(2)	YES		0	
raj_minute	int(2)	YES		0	
raj_second	decimal(12,10)	YES		0.0000000000	
decj_degree	int(2)	YES		0	
decj_minute	int(2)	YES		0	
decj_second	decimal(12,10)	YES		0.0000000000	
dm	decimal(16,10)	YES		NULL	
period	decimal(32,24)	YES		NULL	
bw	decimal(16,6)	YES		NULL	
cal_file_loc	varchar(128)	YES		NULL	
cnfg	varchar(128)	YES		NULL	
data_loc	varchar(128)				
data_type	varchar(16)	YES		NULL	
freq	decimal(32,16)	YES		NULL	
inst	varchar(32)	YES		NULL	
MJD	decimal(32,26)	YES		NULL	
npol	int(6)	YES		NULL	
nchan	int(6)	YES		NULL	
nbin	int(6)	YES		NULL	
nsub	int(6)	YES		NULL	
processing_options	varchar(255)	YES		NULL	
is_calibrated	tinyint(1)	YES		0	
rcvr	varchar(16)	YES		NULL	
site	varchar(8)	YES		NULL	
file_size_bytes	int(32)	YES		NULL	
length	decimal(32,20)	YES		NULL	
obsrvr	varchar(32)	YES		NULL	

Table 2: Fields for the `observations2` table in the mySQL database.

B Proposed “S70” table setup

Field	Type	Null	Key	Default	Extra
file_id	smallint(5) unsigned		PRI	NULL	auto_increment
filename	varchar(30)	YES		NULL	
fileloc	varchar(100)	YES		NULL	
hdrver	varchar(10)	YES		NULL	
survey	varchar(10)	YES		NULL	
src_name	varchar(20)	YES		NULL	
telescope	varchar(16)	YES		NULL	
data_type	varchar(16)	YES		NULL	
raj	varchar(15)	YES		NULL	
decj	varchar(15)	YES		NULL	
rajd	double	YES		NULL	
decjd	double	YES		NULL	
gl	decimal(9,4)	YES		NULL	
gb	decimal(9,4)	YES		NULL	
bmaj	float	YES		NULL	
bmin	float	YES		NULL	
bpa	float	YES		NULL	
date_obs	varchar(30)	YES		NULL	
mjd	double	YES		NULL	
obsfreq	double	YES		NULL	
chanbw	double	YES		NULL	
nchan	int	YES		NULL	
scanlen	double	YES		NULL	
tsamp	double	YES		NULL	
nbits	int	YES		NULL	

Table 3: Fields for the proposed 70cm data table in the mySQL database.