Fig. 1

Fig. 2

1. ADQL Gems

(cf. Fig. 1)

Markus Demleitner
msdemlei@ari.uni-heidelberg.de

(cf. Fig. 2)

- Remote global analysis: Group
- Remote global analysis: Think sets
- Geometry columns
- Deal with epoch differences
- Data discovery and pyVO

In-depth discussion and execution on request.

Sample queries are for http://dc.g-vo.org/tap.

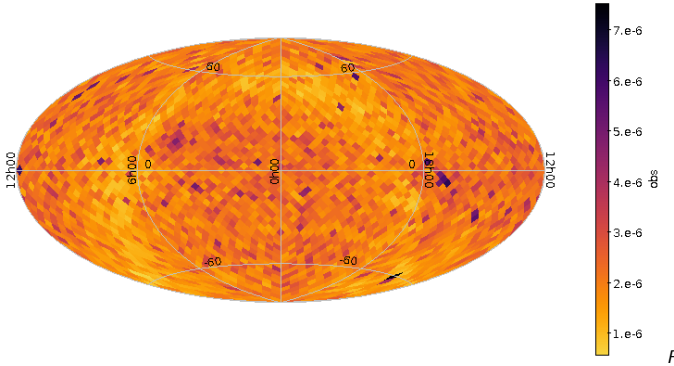

Fig. 3

2. GROUP: Extra nice with healpix

Make healpix maps of arbitrary things without downloading whole catalogs (here mean parallax from Hipparcos):

(cf. Fig. 3)

This image was produced by TOPCAT; recent versions have that in the sky plot.

3. Healpix, Group, Query

The query behind the previous graph:

```
SELECT
  COUNT(*) as n,
  ivo_healpix_index (4, ra, dec) AS hpx,
  AVG(parallax) AS obs
  FROM hipparcos.main
  GROUP BY hpx
```

Remember: Only aggregate functions in the select

The Hipparcos example isn't too exciting; you can do this locally. I've chosen it because it's fast to play around with. You can use the same technique with billion-source catalogs on the remote side, though, and there doing similar tricks locally is a much tougher proposition.

There's a nice piece on this in http://arxiv.org/abs/1611.09190.

Note that ivo_healpix_index is an extension function not yet available on terribly many services.

4. Think Sets

Databases deal with sets. Try to put your problems in set language rather than in array language.

For instance, to find fields with objects "sticking out", you could say:

```
SELECT
    COUNT(*) as n,
    ivo_healpix_index (4, ra, dec) AS hpx,
    MIN(1/parallax)/AVG(1/parallax) AS obs
    FROM hipparcos.main
    WHERE parallax>0
    GROUP BY hpx
```

An example where this is used to find emission lines in CALIFA can be found on http://dc.zah.uni-heidelberg.de/califa/q3/s/info.

Also note how you can write expressions into aggregate functions, and observe that these, in effect, compute values from sets.

5. Geometry Columns

Some tables have columns with geometries in them. You can then use these directly in your relations. Here's an example computing X-ray "color" by the probable source regions of neutrinos:

```
SELECT
        a.id,
        AVG(energy_cor) AS mean_energy,
        COUNT(*)/(ang_error*ang_error)/exposure_time
FROM antares.data AS a
JOIN rosat.photons AS p on
1=CONTAINS(POINT('', p.raj2000, p.dej2000),origin_est)
GROUP BY a.id, ang_error
```


6. Epoch Differences

If catalogs are on different epochs and at least one has proper motions, you can still match them remotely. However, to still exploit spatial indices, make this a two-step process:

```
SELECT
   TOP 30
   *
   FROM ppmxl.main AS m
   JOIN gaia.dr1 AS g
   ON 1=CONTAINS(POINT('ICRS', m.raj2000, m.dej2000),
                 CIRCLE('ICRS', g.raj2000, g.dej2000, 30./3600.))
WHERE 1=CONTAINS(POINT('ICRS',
               m.raj2000+m.pmra*COS(RADIANS(m.dej2000))*15,
               m.dej2000+m.pmde*15),
  CIRCLE('ICRS', g.raj2000, g.dej2000, 0.5/3600.))
```

Without the 30-arcsec inner match, this would be really slow.

For now, don't worry about the "cartesian approximation" for applying proper motions. You'd have to have unusally large epoch differences to make this matter at this precision. But yes, "apply proper motions" is a prime candidate for an ADQL extension function.

7. Data Discovery

"I want proper motions (or radio fluxes) for these objects". There's a form for this at http://dc.g-vo.org/WIRR, but of course you're better off using TAP.

Check the rr schema (here: "TAP tables covering M42 and having proper motions"):

```
SELECT TOP 1 DISTINCT access_url, table_name
FROM rr.interface
NATURAL JOIN rr.capability
NATURAL JOIN rr.res_table
NATURAL JOIN rr.table_column
NATURAL JOIN rr.stc_spatial
WHERE
        standard_id LIKE 'ivo://ivoa.net/std/tap%'
        AND ucd LIKE 'pos.pm%'
        AND 1=CONTAINS(gavo_simbadpoint('M 42'), coverage)
```

If you actually have an interesting data discovery problem, there's an article in Astronomy and Computing that explains the design of RegTAP: 2015A&C....11...91D.

Also note: Right now, this query doesn't actually do what it's supposed to due to deficiencies in the data providers' registy records. So, if you want to do something like that with TAP services in the next year or so, you should use the non-standard GloTS service.

8. Realistically: pyVO

To run a cross-service query, you probably want to use pyVO: VO protocols for python.

```
def get_services_and_tables(regtap_query):
        reg_svc = pyvo.dal.TAPService(REGTAP_ENDPOINT)
        result = reg_svc.run_sync(regtap_query)

        svcs = {}
        for row in result.table:
                svcs.setdefault(row["access_url"], []).append(row["table_name"])
        return svcs.items()

def main():
        recs = []
        for svc_url, tables in get_services_and_tables():
        # in the future: get_services_and_tables(REGTAP_ENDPOINT)
                recs.extend(
                        list(get_rows_for_svc(svc_url, tables)))
```

Full example[1] (needs this helper[2])

9. Thanks.

Grab your copy of the lecture notes at

http://docs.g-vo.org/talks/2017-galaxycoffee.pdf

[1] http://svn.ari.uni-heidelberg.de/svn/edu/trunk/pyvo/query_lots.py
[2] http://svn.ari.uni-heidelberg.de/svn/edu/trunk/pyvo/vohelper.py