Fig. 1

1. MOCs, EPN-TAP, and DaCHS

Markus Demleitner
msdemlei@ari.uni-heidelberg.de

- What's MOCs?
- Where do they fit in EPN-TAP?
- How do I do it?

(cf. Fig. 1)

2. What's a MOC?

MOC = Multi-Order Coverage

- Coverage: A part of a sphere that contains something
- There's no requirement on the shape of the coverage (convexity, connectedness)
- Multi-Order: Let me elaborate.

See http://ivoa.net/documents/MOC for full details.

(cf. Fig. 2)

3. HEALPixes

MOCs are built on HEALPixes, a tesselation of the sphere that can have tiles (well, pixels from here on) of different sizes:

(cf. Fig. 3)

The MOC idea: Where's there's a large area, use large pixels and save on storage. Near a border, use small pixels to represent them well.

A well-thought-out scheme of numbering the pixels also helps to keep the representation compact.

HEALPixes have, in themselves, quite a few desirable properties; in particular, at a given level all have the same size, which is nice for statistics and plotting (see Taylor et al, 2016[1] for inspration).

Also note that expressing coordinates as HEALPixes and then reproject within them lets you avoid ugly artefacts at the poles. That's why Aladin does so much better there than Google Earth last time I looked.

[1] `https://ui.adsabs.harvard.edu/abs/2016arXiv161109190T`

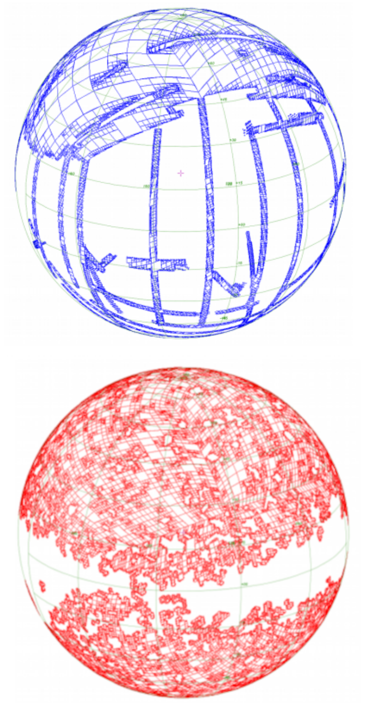

Fig. 2

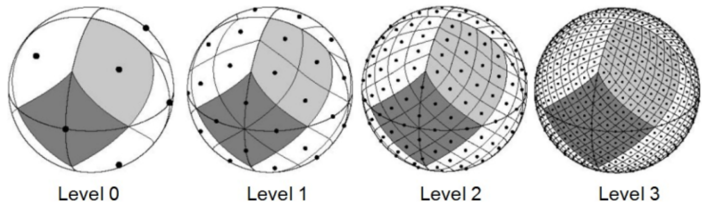

Fig. 3

4. MOC's Pros and Cons

Pro

- They're an algebra with union and intersection – that is: if you join or intersect MOCs, the result is a MOC again. That's not the case with circles or (mostly) polygons.

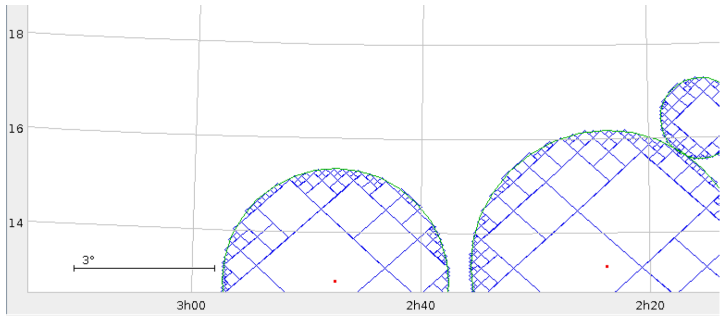

Fig. 4

- Most operations are simple manipulations of a few integers and therefore fast and easy to get right. That is sensational within the field of spherical geometry, which is almost always messy and full of special cases.
- Can easily deal with arbitrary shapes.

Con

- Limited resolution– as opposed to "Circle 12 -30 0.02", where for any point of the sphere you can decide with certainty if it's inside or outside the circle, in the circle's MOC representation there will always be pixels that contain points from both within and outside of the circle.
- For simple geometries, large representation – again, a circle can be described in 24 bytes in double precision; you'd need a *lot* more pixels to do a similarly precise description with a MOC. Of course, that advantage is quickly lost as you get more realistic geometries and precision expectations.

5. HEALPix Rules Of Thumb

At level six, you have resolution of about one degree. Each level up gives you twice the resolution, each level down half the resolution.

6. MOCs in EPNcore

EPNcore defines an `s_region` column that can contain an arbitrary geometry. I've experimentally MOCified the Mars craters (level 10):

(cf. Fig. 4)

7. Example

MOCs are particularly cool if their being an algebra counts.

For instance: Where do two non-trivial craters overlap?

```
WITH overlapping AS (
SELECT
  a.s_region AS r1,
  b.s_region AS r2
FROM mars_craters_moc.epn_core AS a
JOIN mars_craters_moc.epn_core AS b
ON (1=intersects(a.s_region, b.s_region))
WHERE a.granule_uid>b.granule_uid
  AND a.radius>20000 AND b.radius>20000)

SELECT SUM(gavo_mocintersect(r1, r2)) AS ov
FROM overlapping
```

This uses an ADQL 2.1 common table expression ("WITH") – they're cool, and they've been available on DaCHS for a long time now; I'd say it's mostly safe to use them in an EPN-TAP context by now.

Everything in the CTE is standard, and you can do it with standard ADQL geometries. All the magic is in the `SUM(gavo_mocintersect())`.

First, `gavo_mocintersect` is a user defined function that takes two MOCs and returns their intersection (there's also `gavo_mocunion`). The "gavo" prefix lets you guess that for a while we'll only have that in DaCHS. I personally would expect that this will move into an ADQL feature rather than a common UDF. For while, that ought to work.

Then, `SUM` is a normal ADQL aggregate function that just works as (presumably) expected on MOCs: It joins them all together (there's no equivalent of that for intersections yet; if you have a use case for that, let me know). The net effect is that what you get back is a single value, a MOC, defining the shape of "doubly impacted area" if you will.

If you're curious: That shape takes about 170k in its ASCII serialisation and starts with 5/4485 12099 12101-12103 12108-12109 12114 6/17068 17949 17951 23095 38530 43639 43747 45653 46196 47036 47038-47039 47082 48391 48401-48403 48441 48443-48446 48448 48450 48480 48482 7/26 757 923 1384 1625 1749.

8. TOPCAT Plotting the Result

(cf. Fig. 5)

As of May 2020, only beta versions of TOPCAT can plot MOCs from columns. And then you'd need a bleeding-edge pgsphere version on the server side. There is no public service letting you do that just yet. If the Bremen people are fine with it, I can put up the MOCified craters on the Heidelberg service some time in June.

In late 2020, this should all be resolved.

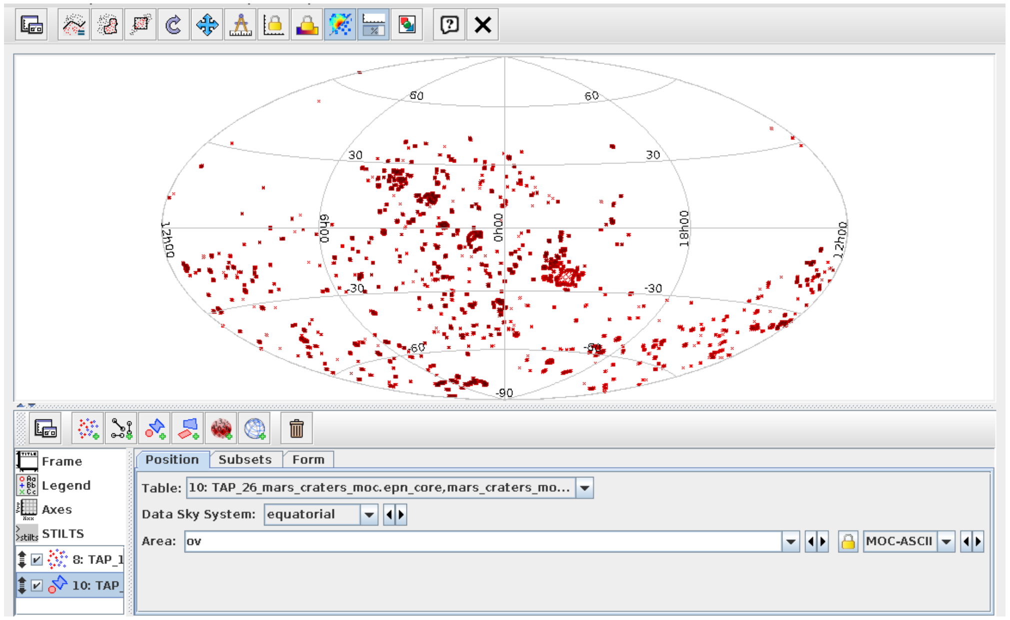

Table: 10: TAP_26_mars_craters_moc.epn_core,mars_craters_mo...

Data Sky System: equatorial

Area: ov MOC-ASCII

Fig. 5

9. Going on

Now, suppose you have a table with surface properties on Mars and would like to see if their distribution is different in our double impact areas. I don't have such a table (and since other EPN-TAP services can't do this yet, I've not bothered to look for one), so I'm using magnitudes in a sky catalogue:

```
select
  avg(mv) as mu,
  sqrt(avg(power(mv, 2))-power(avg(mv),2)) as sig
from arihip.main
join tap_upload.t10
on 1=contains(point(raj2000, dej2000), ov)
```

Here, it's 8.44 ± 1.28 vs. 8.42 ± 1.22 for the whole catalogue. It hence seems the world has some basic sanity and Mars craters are not entangled with bright stars. Oh, and of course you can't work like this with magnitudes, as they're the log of the physical property (flux). If you ever do something like this with mags, do `log10(avg(power(10, mv))` or equivalent. But that would have spoiled my query here.

10. Standards Issues

HEALPix, strictly, is defined only for for (sky) equatorial, Galactic, or ecliptical coordinates.

To be on the safe side someone would some day need to investigate what we ought to do to legally have HEALPixes on planet surfaces.

Meanwhile, as long as people know what they're doing, things will work even without full sanctification.

11. MOCs and DaCHS: The column

If you want your `s_region` to be a MOC, just add, in your table definition mixing in `//epntap2#table-2_0`:

```
<column original="s_region" type="smoc"/>
```

That's it. This, by the way, works in general: If you want to override some attribute of a column coming from the mixin, just add a column element with the original attribute and put in the attributes you want to change.

12. Making MOCs in DaCHS

To fill `s_region`, Mars crater example:

```
<var key="s_region">pgsphere.SCircle.fromDALI([
    @c1min,@c2min,@radius/DEG/(3390.0*1000.0)]
  ).asSMoc(order=10)</var>
```

Or make a polygon and use its asSMoc method. Note, however, that the library underlying this is rather picky with the polygons it accepts; in particular, it can only do convex polygons.

Or get an ASCII MOC from somewhere (e.g., the excellent CDS healpix library) and use

```
<var key="s_region">
  pgsphere.SMoc.fromASCII("8/122-124 342 9/4223")</var>
```

13. In Conclusion

When

- You have complex geometries or
- your users want to do algebra on your s_regions

MOCs are there for you.

You may want to wait a few months before going ahead, though. And you'll need DaCHS2 for most of it.