Fig. 1

1. An ADQL Astrometry Library

Markus Demleitner
msdemlei@ari.uni-heidelberg.de

- ADQL has no (realistic) way to properly apply proper motions – which becomes more troublesome with each year we progress from J2000.0.
- ADQL can no longer transform between reference frames. Well, it really never could, but that's another story.

Here, I'm proposing ADQL user defined functions to address these deficiencies.

(cf. Fig. 1)

2. Proper Motion: Which?

The first question to answer is: "What does it mean to apply a proper motion?" There are at least three options (the queries will run on http://dc.g-vo.org/tap for a while in case you want to play) when you don't know about distances and radial velocities and hence cannot reconstruct the true space motion.

Naive – just add the coordinates; that's blue in the plot:

```
SELECT
  30+0.01/COS(RADIANS(30))*years*1000 AS ra,
  30+0.003*years*1000 AS dec
FROM generate_series(1, 50) AS years
```

Through the tangential plane, which is what DaCHS has done for a while now abusing the `ivo_` prefix; that's red in the plot:

```
SELECT COORD1(p) AS ra, COORD2(p) AS dec FROM (
  SELECT ivo_apply_pm(30, 30, 0.01, 0.003, years*1000) AS p
  FROM generate_series(1, 50) AS years) AS q
```

"Rigourously" by reconstructing the space motion assuming a tangential motion throughout, which is what ESA does and was used, for instance, in the constructions of ARI's Fundamentalkataloge; that's green in the plot:

```
SELECT coord1(p) AS ra, coord2(p) AS dec FROM (
  SELECT ivo_apply_pm(30, 30, 0.01, 0.003,
    CAST('2000-01-01' AS TIMESTAMP),
    CAST((2000+years*1000)||'-01-01' AS TIMESTAMP)) as p
FROM generate_series(1, 50) as years) AS q
```


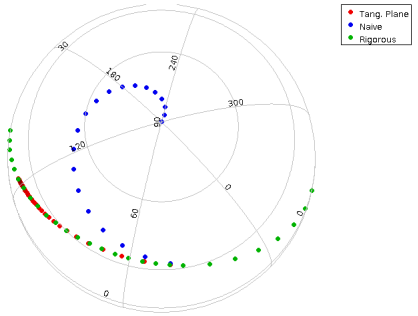

Fig. 2

No, we're not planning to keep `ivo_apply_pm` overloaded in this crazy way. That's just while we're working.

(cf. Fig. 2)

While the naive method clearly makes no sense, it is *not* immediately clear that the rigorous method actually is "right", in particular because at least in the past many catalogues computed their proper motions in the tangential plane.

In the end it's a Bayesian problem (what do we assume for missing information?) that I've not thought about enough to make a call, and thus let's just follow what ESA already does.

3. Calling Convention?

Basic function at ESA:

```
epoch_prop_pos(
  ra DOUBLE, dec DOUBLE, parallax DOUBLE,
  pm_ra DOUBLE, pm_de DOUBLE, radial_velocity DOUBLE,
  ref_epoch DOUBLE, out_epoch DOUBLE) -> POINT
```

where NULLs are interpreted as zero (even for ra and dec). Units used: deg, mas, mas/yr, km/s, epochs in Julian years.

Basic function in DaCHS so far:

```
ivo_apply_pm(
  ra DOUBLE, dec DOUBLE,
  pmra DOUBLE, pmde DOUBLE,
  epdist DOUBLE PRECISION) -> POINT
```

where any NULL argument yields a NULL result. Units: deg, deg/yr, epoch difference in Julian years.

(Apologies again for abusing `ivo_` – this was named long before we agreed on how this ought to be used, and I'll change this UDF as we get on here).

Both agree on returning a spherical point – and it's reasonable to work with that in standard ADQL. Let's do that.

ESA accepts parallax and rv, which are necessary for a rigorous treatment, and given that Gaia has them for many objects, that's a good match for them. Should we still have a variant of the function that can do without them, as that will still be a common use case?

ESA has two parameters for the epoch, where one would, really, do; the first thing the implementation does is take their difference and then forget about them. Me, I almost always prefer

fewer arguments in functions. But then: it *is* a little more explicit for write `..., 1991.25, 2023.432)` than to write `..., 32.182)`.

One *might* think about allowing timestamps to specify epochs. But given you cannot compute with them in ADQL, that could only be an extra, and for that, I don't see a major gain (corollary: don't use timestamps in your TAP tables).

Units: ESA has the units as they are in gaia_source. And most astronomers like them. It *would* be a bit nicer to go a bit further into the decimal age and have degrees (or, even better, rad!) everywhere, but experience shows people aren't wild about consistency...

NULL handling: I'm sure this should return NULL for missing ra and dec. Assuming 0 for missing pmra and pmdec: This *might* produce fake science, but it also makes the thing a lot easier to deal with. Assuming 0 for missing RV and parallax: since the effects are rather minute in practice and most catalogues just don't know about them, I'd say that's reasonable.

4. Six-parameter Transform

ESA also has the following function:

```
epoch_prop(
  ra DOUBLE, dec DOUBLE, parallax DOUBLE,
  pm_ra DOUBLE, pm_de DOUBLE, radial_velocity DOUBLE,
  ref_epoch DOUBLE, out_epoch DOUBLE) -> DOUBLE[6]
```

This is like epoch_prop_pos but returns an array that also has PM, RV, and parallax at the new epoch.

While it's clear that this functionality is necessary, it is perhaps less clear it has to be available in the database.

5. 5-Parameter Transform with Errors

ESA also has:

```
epoch_prop_error(
  ra DOUBLE, dec DOUBLE,
  pmra DOUBLE, pmde DOUBLE,
  e_plx DOUBLE,
  e_pmra DOUBLE, e_pmdec DOUBLE,
  e_vr DOUBLE,
  de_ra DOUBLE, plx_ra DOUBLE, plx_de DOUBLE,
  pmra_ra DOUBLE, pmra_de DOUBLE, pmra_plx DOUBLE,
  pmde_ra DOUBLE, pmde_de DOUBLE, pmde_plx DOUBLE, pmde_pmra DOUBLE,
  e_dec DOUBLE, e_ra DOUBLE, rv DOUBLE, plx DOUBLE) -> DOUBLE[21]
```

I'd raise a TooManyParametersException here. To me, the sheer length of the parameter list suggests we should explicitly introduce the covariance matrix and pass that as a single parameter. And I'm not wild about stuffing all the numbers into a single array when returning, either. Hence, I'd say this should not be part of the first batch of astrometry UDFs. Let's wait and see – and perhaps introduce types to simplify dealing with such beasts.

There's also the question of NULL handling here. ESA's docs say "An exception is raised for input stars without a five parameters astrometric solution"; I have not investigated that more closely.

6. 6-Parameter Covariance Transform

Finally, ESA has:

```
epoch_prop_covariance(
  ra DOUBLE, parallax DOUBLE,
  pm_ra DOUBLE, pm_de DOUBLE, radial_velocity DOUBLE,
  e_ra_deg DOUBLE, e_de_deg DOUBLE, e_plx DOUBLE,
  e_pm_ra DOUBLE, e_pm_de DOUBLE, e_rv DOUBLE,
  de_ra DOUBLE, plx_ra DOUBLE, plx_de DOUBLE,
  pm_ra_ra DOUBLE, pm_ra_de DOUBLE, pm_ra_plx DOUBLE,
  pm_de_ra DOUBLE, pm_de_de DOUBLE, pm_de_plx DOUBLE,
  pm_de_pm_ra DOUBLE,
  ref_epoch DOUBLE, out_epoch DOUBLE, dec DOUBLE)
  -> DOUBLE[6][6]
```

I'd probably hide that behind whatever we come up of epoch_prop_error. Well, and there's the somewhat odd parameter sequences on top.

7. Frame Transformation

Do naive, 1st order transforms between reference frames.

`CONTAINS(POINT('ICRS'), CIRCLE('GALACTIC'))` should have done this automatically, but that's a can of worms we're closing in ADQL 2.1.

DaCHS has:

```
gavo_transform(
  from_sys TEXT, to_sys TEXT,
  geo GEOMETRY) -> GEOMETRY
```

There are a lot fewer free parameters on this than with epoch propagation, but...

8. Frame Transformation Tunables

- Name: "transform" is a bit generic
- Frame identifiers: probably from http://www.ivoa.net/rdf/refframe
- ...but which of them does a concrete service support? Does this need to be machine-readable?
- What about frames that need an equinox (e.g., FK4 for 1975.0)?
- Only geometries? Or should we have a variant with long and lat, too?
- All geometries or just points?

Implementation feedback: Based on pgsphere, this was straightforward – you just define the rotation and apply it to whatever geometry you have.

This *would* become messy if you wanted to actually correct for rotation between frames (significant between FK4 and FK5), the relativistic FK4 corrections, "wrinkles" in the frames, etc. I'd argue that's beyond what people can reasonably expect from a database.

9. Proposed Plan

- Adopt ESA's `epoch_prop_pos` with improved NULL behaviour into the UDF catalogue.
- Think about a simplified version without parallax and RV (but live with two epochs).
- Push epoch propagation code with covariance matrix math into pgsphere (an alternative would be to use pgsphere's built-in rotation to just have 4-parameter epoch propagation; but I'd say that's not worth the effort when we'll want the real thing one day anyway).
- Adopt `gavo_transform` into the UDF catalogue – **needs a second implementation!**
- Postpone everything else to some later date.

Are you in?