Fig. 1

Fig. 2

Fig. 3

1. ADQL 2.1: DaCHS Implementation

(cf. Fig. 1)

Markus Demleitner
msdemlei@ari.uni-heidelberg.de

(cf. Fig. 2)

ADQL 2.1 is in PR (yes!).

I've implemented most of it in DaCHS.

- Stuff done I'm happy with
- Stuff done I'm unhappy with
- Stuff not done I'm unhappy with

(cf. Fig. 3)

2. Happy: Geometries

- `POINT(ra, dec)` rather than `POINT('ICRS', ra, dec)`: polymorphism was a bit tricky (`POLYGON!`). Otherwise: great, in particular for 2.1-only implementations
- `CIRCLE(POINT(ra, dec), radius)` – a bit involved because of things like `CIRCLE(center_col, radius)`, but all worth it.
- `DISTANCE(ra1, dec1, ra2, dec2)<radius` instead of `1=CONTAINS(POINT, CIRCLE)` – good move.
- `DISTANCE(p1, p2)<radius` tricky index-wise if using p2 from an upload, but DaCHS will scupper q3c some day anyway.

3. Happy: CTEs

Common Table Expressions are great.

I've put one into my server's examples:

```
WITH sample AS (
        SELECT * FROM gdr2mock.main
        WHERE distance(ra, dec, 66.73, 75.87)<2)
SELECT ROUND(age*10)/10 as bin,
        avg(ra) as m_ra, max(1/parallax) as x_px
FROM sample
GROUP BY bin
```

Except: I'm forcing query_name to be regular_identifier – please!

By the way, these are surprisingly simple in implementation. Just go for it if your DB can do them!

4. Happy: IN_UNIT

DaCHS has had `IN_UNIT(pmra, 'deg/yr')` for quite some time and people love it.

Join us now! (Though it may be a bit of work if you don't do unit inference yet)

5. Happy: Misc

- `LOWER(s)` – straightforward, frequently useful, hard to imagine a DBM that would give trouble with it.
- `OFFSET` – straightforward, grammar looks ok.

6. Ah well: CAST, TIMESTAMP

`CAST('123', INTEGER)` translates easily, **but**

- Grammar for CAST is missing
- Current DaCHS grammar for CAST doesn't have SQL length, array. . .
- The grammar needs some way of saying "type"
- `CAST(x, TIMESTAMP)` is the same as `TIMESTAMP(x)` – that's ugly

Let's not have `TIMESTAMP(x)`

7. Ah well: ILIKE

Case-insensitive string matching sounds good:

`instrument ILIKE 'hst%'`

– but RegTAP already defines

`ivo_nocasematch(instrument, 'hst%')`.

Having both sucks, and we should drop one.

(RegTAP 1.1 is in WD. . .)

DaCHS doesn't have ILIKE yet. Speak out!

8. Ah well: set operations

```
SELECT a,b FROM table1
UNION SELECT a,b FROM table2
EXCEPT (
  SELECT a,b FROM table3
  INTERSECT
  SELECT a,b FROM table4)
```

They are useful, but I fear the ADQL 2.1 grammar might be bad.

DaCHS uses a different grammar, so it *does not* count as implementation at this point.

9. Resist, Refuse: Misc

- `BOX(p, width, height)` – close to unimplementable on top of pgsphere with questionable utility. Let'd deprecate BOX altogether.
- `INTERVAL` type – severely underspecified right now, nobody seems to work on it

10. R,R: boolean functions

The current grammar lets you write

`WHERE bool_fct(a)`

and

`WHERE True`

but not, I think,

```
SELECT * FROM
  (SELECT bool_fct(a) AS x FROM table) as q
WHERE x
```

This is all too confusing.

Let's not have more than `x=True` or `f(a)=True` lest things get out of hand.

11. R,R: bitwise operators

The spec wants bitwise operations as operators: `flags&8=8`.

DaCHS does them as functions: `BITWISE_AND(flags,8)=8`.

Operators are a bit more readable, yes, but the grammar is an order of magnitude harder to write, and what's currently in the spec almost certainly is not what we want.

Let's have functions!

12. Other Issues

- The PR references the STC-S note for REGION argument – that's underspecified and overkill. Let's copy the TAP-1.0 appendix.
- The grammar doesn't reflect all the text says (look at the CRICLE grammar)
- Do we want geometry-values user-defined functions?
  (`gavo_move_pm` returning a POINT is fairly popular to change epochs)

13. In Conclusion

Blog post on this with a view to interested astronomers:

https://blog.g-vo.org/speak-out-on-adql-2-1/

Offer: I'll volunteer for doing edits we find consensus on.

Thanks!