Fig. 1

Fig. 2

1. ADQL extensions implementation

(cf. Fig. 1)

Markus Demleitner
msdemlei@ari.uni-heidelberg.de

(cf. Fig. 2)

This is a brief report on the implementation of the following new ADQL features that have been proposed at one time or another.

- set operations
- CROSSMATCH function
- user-defined geometry functions
- UCDCOL macro
- IN_UNIT macro
- string functions

Play with the new features at http://dc.g-vo.org/tap[1].

2. Set Operations

Plan: Include UNION, INTERSECT, EXCEPT operators.

Trouble:

- Odd SQL92 grammar
- Interesting rules on merging columns
- uneven backend implementations

Grammar issue: SQL92 grammar appears to forbid things like

`select * from t1 union select * from t2`

DBs like Postgres and SQLServer want exactly this. In the end, I went for an adaption of postgres' grammar.

However, as Postgres as LIMIT instead of ADQL's TOP, and it only supports a global limit in constructs as above, that's not totally straightforward either.

For all I can see, SQL92 set operations are not what we want. However, I admit that having to come up with a sanitised grammar for set operations makes them a whole lot less attractive.

Who's going to join me in looking into this? Maybe I'm missing something trivial.

[1] `http://dc.g-vo.org/tap`

3. CROSSMATCH

Plan: let people write `1=CROSSMATCH(ra1, dec1, ra2, dec2, sr)` instead of mess with CIRCLE and POINT..

Trouble: The function is straightforward. However, users might expect a function "closest within match limit" rather than "all pairs up to sr".

That, however, breaks the relational model and is fairly messy to implement. Do we still want it?

4. Geometry UDFs

Plan: let operators define user defined functions returning geometry values. Users could write `select ivo_apply_pm(ra, dec, pmra, pmdec, 2015.4-cat_epoch ...` and get back a usable POINT.

Trouble: None. The grammar from the TAP Implementation Note works as expected.

5. UCDCOL

Plan: Let people write `UCDCOL('pos.eq.ra;meta.main')` and have that replaced with the first column with the UCD.

Trouble:

- We need to change the columnReference production in the grammar, which is so low-level that unintended consequences may result.
- Tables to pull columns from aren't available when UCDCOL is parsed
- Didn't feel right in implementation.
- Probably not all that helpful in writing generic queries. You'd at least still have to fumble with table names.

6. IN_UNIT

Plan: Let people write things like

$$\mathtt{dec} + \mathtt{IN\_UNIT}(\mathtt{pmde}, '\mathtt{deg/yr}') * 24.4$$

or even

$$\mathtt{IN\_UNIT}(\mathtt{sqrt}(\mathtt{pmra} * \mathtt{pmra} + \mathtt{pmde} * \mathtt{pmde}), '\mathtt{mas/yr}').$$

Trouble: Very little. Of course, I've been unit-annotating expressions before that. If I had to do it just to support this, I'd curse the jerk that came up with it.

7. String operations

Plan: give people a chance to do case-insensitive string comparisons.

Solutions: I did sql92-style UPPER and LOWER, not yet ILIKE (there's `ivo_nocasematch` from RegTAP, though).

Trouble:

- Grammar is missing in Note/PR –
```
<fold> ::= { UPPER | LOWER } <left paren>
  <character value expression> <right paren>
<string_value_function> ::= <fold>
  | <string_geometry_function> | <user_defined_function>
```
- If LOWER, why not UPPER?
- Do we want any of the others? There's SUBSTRING, CONVERT, TRANSLATE, TRIM in sql92. Of course, they're not necessary to scratch the original itch. . .

8. Conclusions

- Set operations are a larger pain than I had expected, but I still believe we need them. They need work, though, the SQL92 grammar in the Note probably is not what we want.
- UCDCOL is probably not worth it.
- CROSSMATCH needs thought as to how we can best bridge the gap between astronomer's expectations and relational calculus.
- The rest is reasonable.