Fig. 1

Fig. 2

1. Whither ADQL

(vgl. Fig. 1)

Markus Demleitner
msdemlei@ari.uni-heidelberg.de

(vgl. Fig. 2)

- Bugs
- Wishlist
- Vision
- Validation

2. Bugs and Misfeatures

(These come from TAP implementation notes)

- Missing language on the separator nonterminal. A naive implementation of the grammar will only allow comments between parts of split-up string literals.
- Reference system argument on Geometries: Can we deprecate it? I've yet to meet a single person who likes it, I've yet to see convincing use cases for it, and the behavior is only vaguely defined. However, it's yet another of these little things that make implementing ADQL painful.
- Decay of INTERSECTS to CONTAINS: Can we deprecate it? This one again means implementation effort, since you'll have to keep track of the argument types. This is the only feature in current ADQL that positively requires type-annotation of inner nodes of the parse tree (at least on top of pgSphere).
- Lack of a type system. The table listing valid types in the TAP standard should have been in the ADQL document, complete with suggested VOTable types. This would also be a good opportunity to outfit ADQL with booleans...

3. Wishlist

- Simple crossmatch function: `t1 join t2 on (1=crossmatch(t1.ra, t1.dec, t2.ra, t2.dec, 0.001)`. The thing with contains and circle is all nice and flexible, but the default circular crossmatch is so frequent we should support it with a simple function.
- User defined geometric (and maybe string)-valued functions. I'd like to have a function to, e.g., apply proper motion to positions. Right now, this doesn't (really) work since user defined functions are always numeric: `select move_pm(pos, pm, -22)`
- `ILIKE` or `LOWER` – right now, there's no portable way to do case-insenstive comparisons/matching in ADQL.
- `UNION` — well, it's one of the fundamental operations of relational algebra. There's also no way to concatenate relations otherwise.
- Booleans – though we probably cannot fix CONTAINS and INTERSECTS any more, but within the type system we should definitely mention booleans. I'm not aware of a major RDMS that would be ruled out as a basis of ADQL because of that.

4. Units and UCDs

A big step forward for an astro*physical* query language would be unit-awareness. A fairly easy addition could be a unit cast, e.g., by a function
`in_unit(expression, unit_string) -> numeric`
that would fail if the service doesn't know the unit of expression or cannot convert between the two units.

In reality, to be able to come up with valid units in the resulting VOTables, ADQL would need to get numeric literals with units. That requires new syntax, and we'd need to carefully weigh the benefits against adding syntax.

Another cool feature in data discovery could be column selection by UCD, e.g.,
`SELECT [[pos.eq.ra*]], [[pos.eq.dec*]], [[phot.mag*]]...`
– but that wants more thought.

5. Visions

If you're looking for a topic for a grant application: ADQL 3.0 should, in my view:

- support subqueries syntactically, like functions, to facilitate factoring,
- still be text-based, but easy enough to parse and interpret that graphical front-ends are feasible,
- more explicitely support the idea of pipelines (in the Unix sense, except that joining them would be a much more common operation) – people seem to be quite comfortable with that metaphor,,
- lean towards declarative constructs ("my problem is" rather than "do this, then that").
- have a sqlite-based implementation from start.
- still be non-Turing complete – actually, it should be provably possibly to transform any valid ADQL 3.0 program to SQL.

I frankly have no clue what such a beast would look like, except that I think some rather interesting work has been done within AstroWISE.

6. Validation

There are now a number of ADQL implemenations out there, and more are coming. ADQL is complex, and the grammar has a few pitfalls.

Thus: We need a validation suite. A while ago, I made an attempt at starting one two years ago, which didn't take off but still is at

http://svn.ari.uni-heidelberg.de/svn/gavo/ADQLValidation/trunk/

Bold coworkers wanted to take that up again.