ADQL

Markus Demleitner *(msdemlei@ari.uni-heidelberg.de)*

Agenda

- ORDER
- WHERE
- SELECT
- REGION
- FROM

First steps in the

Astronomical

Data

Query

Language

Why bother?

Moving multi-terabyte or petabyte data sets is not an option in astronomy. Even less when you want two of them.

Thus: You will have to move your smarts to where some data happens to be. In addition to smart planning (so only the data actually required gets transferred), you need a language to express your smarts. It's not going to be FORTRAN. Maybe it's going to be ADQL.

Let me play

To learn what ADQL is, see the service info. You will find important information for seasoned ADQL users there as well.

ADQL query

```
SELECT TOP 10 Designation, Inclination/H AS score
FROM mpc.mpcorb
ORDER BY score
```

A query in the Astronomical Data Query Language

Timeout after [s]: 50

Seconds until the query is aborted. If you find yourself having to raise this beyond 200 or so, please contact the GAVO staff for hints on how to optimize your query

Output format HTML

Go

Relational Algebra

At the basis of relational data bases is the relational algebra, an algebra on sets of tuples ("relations") defining six operators:

- unary *select*
- unary *project*
- unary *rename*
- binary *cartesian product*
- binary *union*
- binary *set difference*

Good News: You don't *need* to know any of this.

SELECT for real

ADQL defines just one statement, the `SELECT` statement, which lets you write down expressions of relational algebra. Roughly, it looks like this:

`SELECT` [`TOP` *setLimit*] *selectList* `FROM` *fromClause* [`WHERE` *conditions*] [`GROUP BY` *columns*] [`ORDER BY` *columns*]

TOP

setLimit: just an integer giving how many rows you want returned. ▷1▷2

ORDER BY

columns: a list of column names (or expressions). ▷3▷4▷5

SELECT: what?

select list

The select list has column names or expressions involving columns.

SQL expressions are not very different from those of other programming languages. ▷6

Use `COUNT(*)` to figure out how many items there are. ▷7

WHERE clause

Behind the WHERE is a logical expression; these are similar to other languages as well, with operators `AND`, `OR`, and `NOT`. ▷8

SELECT: Whence

The tricky point in ADQL is the `FROM` clause. So far, we had a single table. Things get interesting when you add more tables: JOIN. ▷9

`JOIN` is a combination of cartesian product and a select. When you write

```
measurements JOIN stations USING (stationid)
```

you get the cartesian product of the measurement and stations tables but only retain the rows in which stationid is the same. ▷10

If your join criteria are more complex, you can join `ON`:

```
dmubin LEFT OUTER JOIN rave ON
  (dmubin.mv BETWEEN rave.imag-1 AND rave.imag+1)
```

▷11

SELECT: Grouping

For histogram-like functionality, you can compute factor sets, i.e., subsets that have identical values for one or more columns, and you can compute aggregate functions for them. ▷12

ADQL's aggregate (set) functions include `AVG`, `MAX`, `MIN`, `SUM`, `COUNT`.

For simple `GROUP` applications, you can shortcut using `DISTINCT` (which basically computes the "domain"). ▷13

Geometries

The main extension of ADQL wrt SQL is addition of "geometric" functions.

Keep the crossmatch pattern somewhere handy (everything is in degrees):

```
1=CONTAINS(
  POINT('ICRS', bigcatRA, bigcatDE),
  CIRCLE('ICRS', smallcatRA, smallcatDE, matchradius))
```

▷14

Some sites have extra features in the `REGION` construct. On my site, you can, e.g., say `REGION('simbad <obj>')`.

▷15

Where do we go from here?

- Get the lecture notes for this talk from http://www.g-vo.org and play around.
- Read up on SQL – there a some ok books out there and many bad ones. The ⟨PostgreSQL docs⟩ are nice, too.
- Read the service info of the DC's ADQL service; there's going to be news there now and then.
- **The Future:** TAP.