# ADQL Reference

## Select Clause

```
SELECT [DISTINCT|ALL] [TOP n]
  <select list>
FROM <table expression>
[WHERE <conditions>]
[GROUP BY <column> {, <column>}]
[HAVING <conditions>]
[ORDER BY <column> {, <column>}  [ASC|DESC]]
```

## ADQL Functions

### Chopping numbers

`ABS(x)`, `CEILING(x)`, `FLOOR(x)`,
`ROUND(x[, places])`, `TRUNCATE(x)`

### Transcendental functions

`EXP(x)`, `LOG(x)`, `LOG10(x)`, `POWER(x, y)`, `SQRT(x)`,
`ACOS(x)`, `ASIN(x)`, `ATAN(x)`, `ATAN2(y, x)`, `COS(x)`,
`COT(x)`, `SIN(x)`, `TAN(x)`

### Other

`PI()`, `DEGREES(x)`, `RADIANS(x)`
`MOD(numerator, denominator)`
*-- remainder on integer division*
`RAND()`, `RAND(seed)` *-- random numbers*

## ADQL Aggregate Functions

These are functions taking sets of rows, either the total result set or from `GROUP`:

`COUNT`, `MAX`, `MIN`, `SUM`, `AVG`

Special case:

`SELECT COUNT(*) FROM table`

counts the rows.

## ADQL Geometry Functions

`<csys>`, the coordinate system, should normally just be the empty string (`''`). All angles are in degrees.

`AREA(<geometry value>)`

`BOX(<csys>, ra, dec, width, height)`

`CENTROID(<geometry value>)`

`CIRCLE(<csys>, ra, dec, radius)`

`CONTAINS(<geometry value>, <geometry value>)` *– returns 0 or 1*

`COORD1(<point or such>)`

`COORD2(<point or such>)`

`DISTANCE(<point or such>, <point or such>)`
*– in degrees*

`INTERSECTS(<geometry value>, <geometry value>)`
*– returns 0 or 1*

`POINT(<csys>, ra, dec)`

`POLYGON(<csys>, ra1, dec1, ra2, dec2 {, ran, decn})`

## ADQL Predicates

Predicates are expressions you can use in `WHERE` clauses.
The "usual" math comparisons work:

`=, !=, <=, >=, <, >`

`a [NOT] BETWEEN x AND y`

`EXISTS (subquery)`

`a IS [NOT] NULL`

Caution: stuff like

`a=NULL`, `a!=NULL`, or `a>NULL`

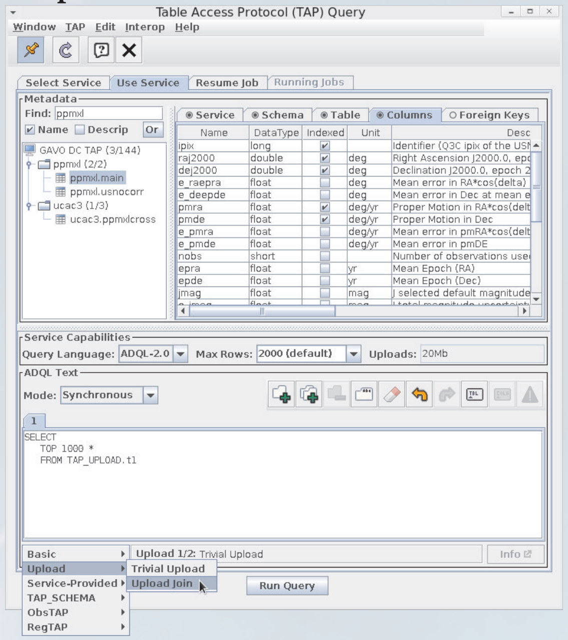
is always false!

## Introduction

This reference card is about the IVOA Astronomical Data Query Language ADQL, a dialect of SQL for querying astronomical databases understood across many servers. ADQL is sent to the servers using the table access protocol TAP.
A fairly gentle introduction to both topics is available at http://docs.g-vo.org/adql.

## TAP Matters

### TAP upload in TOPCAT ...



Find: ppmxl
Name [✓] Descrip [ ] Or

GAVO DC TAP (3/144)
ppmxl (2/2)
ppmxl.main
ppmxl.usnocorr
ucac3 (1/3)
ucac3.ppmxlcross

| Name | DataType | Indexed | Unit | Desc |
|---|---|---|---|---|
| ipix | long | ✓ | | Identifier (Q3C ipix of the USN |
| raj2000 | double | ✓ | deg | Right Ascension J2000.0, epo |
| dej2000 | double | ✓ | deg | Declination J2000.0, epoch 2 |
| e_raepra | float | | deg | Mean error in RA*cos(delta) |
| e_deepde | float | | deg | Mean error in Dec at mean e |
| pmra | float | ✓ | deg/yr | Proper Motion in RA*cos(delt |
| pmde | float | ✓ | deg/yr | Proper Motion in Dec |
| e_pmra | float | | deg/yr | Mean error in pmRA*cos(delt |
| e_pmde | float | | deg/yr | Mean error in pmDE |
| nobs | short | | | Number of observations use |
| epra | float | | yr | Mean Epoch (RA) |
| epde | float | | yr | Mean Epoch (Dec) |
| jmag | float | | mag | J selected default magnitude |

Service Capabilities
Query Language: ADQL-2.0 Max Rows: 2000 (default) Uploads: 20Mb

ADQL Text
Mode: Synchronous

```
SELECT
  TOP 1000 *
  FROM TAP_UPLOAD.t1
```

### ... and in scripts

```
stilts tapquery
  tapurl='http://dc.zah.uni-heidelberg.de/tap'
  adql="SELECT
  TOP 1000 *
  FROM TAP_UPLOAD.t1"
```

## Sample Queries

### Basic Query

A one-table query showing off many frequently-used features.

```
SELECT TOP 10
 POWER(10, alfa_Fe) AS ppress,
 SQRT(SQUARE(e_pmde)+SQUARE(e_pmra))
  AS errTot
FROM rave.main
WHERE  obsDate>'2005-02-02'
 AND imag<12
 AND ABS(rv)>100
ORDER BY ppress
```

### Grouping/Histograms

Here, we make a histogram by visual magnitude and compute color averages for each bin.

```
SELECT
 COUNT(*) AS n,
 ROUND(mv) AS bin,
 AVG(color) AS colav
FROM dmubin.main
GROUP BY bin
ORDER BY bin
```

### Subqueries

Used here to try a query with a subset of a large table; also note how we're extracting digits from a compound flag here.

```
SELECT
  COUNT(*)
FROM (
  SELECT TOP 4000 *
  FROM arigfh.id) AS q
WHERE 4=MOD(q.decflags/10000, 10))
```

Created 2012 by the GAVO project, in particular Kristin Riebe and Markus Demleitner. GAVO is Germany's contribution to the Virtual Observatory effort, funded by Federal Ministry of Education and Research. For details see http://www.g-vo.org.

## Joining Tables

Join with `USING`

*(join tables by giving the names of the columns that must match)*

```
SELECT TOP 10 lat, long, flux
FROM lightmeter.measurements
JOIN lightmeter.stations
USING (stationid)
```

Join with `ON`

*(give a boolean expression; here's a crossmatch using ADQL geometries, matching objects from ppmxl.main to those in rave with a radius of 1.5 arcseconds)*

```
SELECT TOP 5
 rv, e_rv,
 p.raj2000, p.dej2000,
 p.pmRA, p.pmDE
FROM ppmxl.main AS p
 JOIN rave.main AS rave
  ON 1=CONTAINS(
   POINT('', rave.raj2000, rave.dej2000),
   CIRCLE('', p.raj2000, p.dej2000,
    1.5/3600.))
```

NATURAL join

*(use all matching names; this query will give you TAP services giving columns with a certain UCD in tables with a certain keyword)*

```
SELECT ivoid, access_url, name, ucd,
description
FROM rr.capability
 NATURAL JOIN rr.interface
 NATURAL JOIN rr.table_column
 NATURAL JOIN rr.res_table
WHERE standard_id='ivo://ivoa.net/std/TAP'
AND 1=ivo_hasword(table_description,
'quasar')  AND ucd='phot.mag;em.opt.V'
```

Using EXIST

*(this filters all objects present in a second table)*

```
SELECT * FROM ppmxl.main AS q
WHERE NOT EXISTS (
 SELECT * FROM dmubin.main AS d
 WHERE 1=CONTAINS(
  POINT('', d.raj2000, d.dej2000),
  CIRCLE('', p.raj2000, p.dej2000, 0.001)))
```

## Common Obscore columns

The table `ivoa.obscore` describes observations ("datasets") in a generic way. Commonly used columns in that table include:

`dataproduct_type` -- image, cube, spectrum, sed, timeseries...

`obs_publisher_did` -- a VO-unique identifier for the dataset

`access_url` -- where to get the data

`target_name` -- what did they want to observe?

`s_ra, s_dec` -- ICRS center of observation

`s_region` -- (sometimes) an ADQL geometry of sky area covered

`t_min, t_max, t_exptime` -- time covered (MJD), exposure time (s)

`em_min, em_max` -- waveband covered (in meters)

`o_ucd` -- UCD for the observable

`facility_name, instrument_name` -- where did the dataset come from?

## Important TAP_SCHEMA tables

`tables` – `table_name`, `description` tell you what tables there are

`columns` – `column_name`, `description`, `ucd`, `unit`; also check for `indexed`

`keys` – `from_table`, `target_table` give you `key_ids` for foreign keys ("links") between tables. The actual columns that are part of the foreign key are in `key_columns`.