SQL (and ADQL and TAP)

Mark Taylor

Astro Dev Group

3 May 2024

$Id: sql.tex,v 1.16 2024/05/03 11:54:24 mbt Exp $

SQL

Structured Query Language

- Queries RDBMS (Relational DataBase Management Systems)
  - These store/manage **tables**
- Mature/old — since 1970s/80s
- Industry standard
  - but not really standard - everybody does it a bit differently
    - PostgreSQL, SQLite, MySQL/MariaDB, SQL Server, Oracle, DB2, ...
  - standards document is available (e.g. ISO/IEC 9075:2023) — but expensive
- Robust
- Transactional
- Not really a programming language
  - Not Turing-complete
  - Specify *what you want to see*, not *how to get it*
- Based on Relational Algebra

SQLite Example

```
% sqlite3 astro.sqlite
sqlite> CREATE TABLE messier (name TEXT, id INTEGER PRIMARY KEY, ngc INT, con TEXT, type INT, ra REAL, dec REAL, bmag REAL);
sqlite> .mode csv
sqlite> .import messier.csv messier
sqlite> .schema messier
CREATE TABLE messier (name TEXT, id INT, ngc INT, con TEXT, type INT, ra REAL, dec REAL, bmag REAL);
sqlite> .mode column
sqlite> .headers on
sqlite> SELECT * FROM messier LIMIT 3;
name        id          ngc         con         type        ra          dec         bmag
----------  ----------  ----------  ----------  ----------  ----------  ----------  ----------
M1          1           1952        Tau         9           83.50208    22.016666   8.4
M2          2           7089        Aqr         2           323.25208   -0.8166666  6.5
M3          3           5272        CVn         2           205.50084   28.383333   6.2
sqlite> SELECT name, con FROM messier WHERE con = 'Aqr';
name        con
----------  ----------
M2          Aqr
M72         Aqr
M73         Aqr
sqlite> .headers off
sqlite> SELECT COUNT(*) FROM messier;
110
sqlite> BEGIN TRANSACTION;
sqlite> DELETE FROM messier WHERE id>50;
sqlite> SELECT COUNT(*) FROM messier;
50
sqlite> ROLLBACK;
sqlite> SELECT COUNT(*) FROM messier;
110
```

You can interact with it programmatically as well

Why use SQL?

RDBMS/SQL is a good solution for:

- Persisting large amounts of tabular data
- Multiple tables linked by complicated schemas
- Efficient data access
- Transactional access to guarantee database integrity
- User access controls

This is quite heavy duty for local data

- Often there's a simpler solution (CSV, plain text, JSON, ...)

But it makes sense if you're a large data archive storing observational/survey/simulation data

- To serve users you still need some layer between the DB and the user

SQL in Astronomy

- 1990s: SQL is too complicated, astronomers won't learn it
- 2001+: SDSS data releases
  - Main CASJobs interface was SQL `SELECT` statements from a web page
  - Astronomers wanted to use SDSS data ...
  - ... so they learned SQL
- 2000s: Various other data archives had similar-but-different SQL web query forms
- 2008–2010: Virtual Observatory ADQL and TAP standards published
- Now: 100+ registered TAP/ADQL services

ADQL and TAP

ADQL: Astronomical Data Query Language

- Just a standard, restricted dialect of SQL
- All TAP services use it
  - ▷ Services translate locally from ADQL to PostgreSQL, MySQL, SQL Server, ...
  - ▷ So users don't need to worry about (most) server implementation details
- Provides some extensions useful for astronomy
  - ▷ Mostly to do with sky geometry: `CIRCLE`, `POLYGON`, `POINT`, `DISTANCE`, `CONTAINS`, ...
- Only one command: `SELECT`
  - ▷ Enough for making queries
  - ▷ No attempt to implement DB management, table modification, transactions, ...

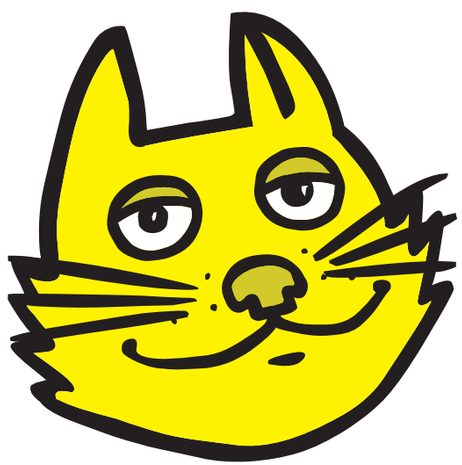
TAP: Table Access Protocol

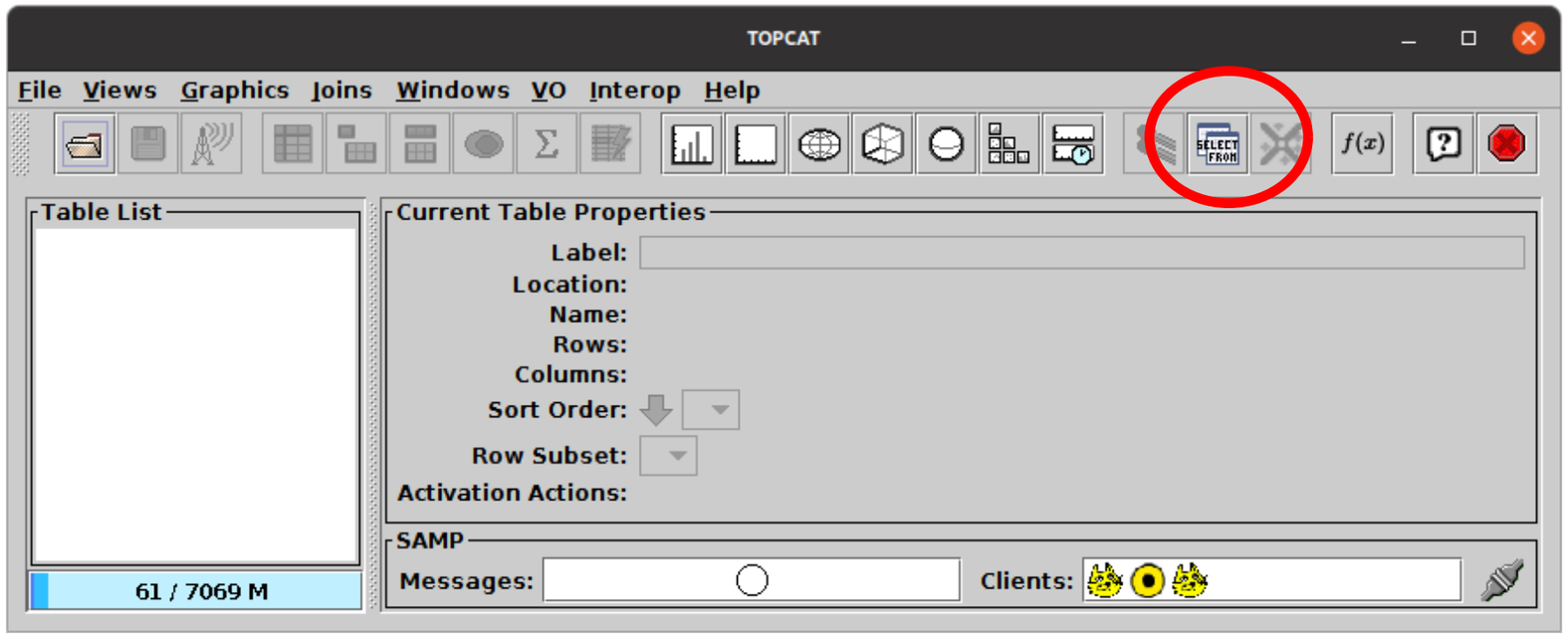
- Standard way for clients (e.g. Python, TOPCAT, web browser) to talk to databases over HTTP:
  - ▷ Submit ADQL queries (synchronous or asynchronous)
  - ▷ Retrieve results
  - ▷ Enquire about table metadata (what tables, what columns, details of both) and service capabilities (row limits etc)
- The details are kinda complicated ... but you mostly don't need to know/worry

TOPCAT and TAP

TOPCAT is one way to talk to TAP services (see `http://www.starlink.ac.uk/topcat/`)

- Run TOPCAT
  - Java + Jar file:
    - Download http://www.starlink.ac.uk/topcat/topcat-full.jar
    - Run `java -jar topcat-full.jar`
  - MacOS and homebrew:
    - `brew install --cask topcat --no-quarantine`
    - Run **TOPCAT** application
- Open TAP window
  - Hit TAP button in toolbar

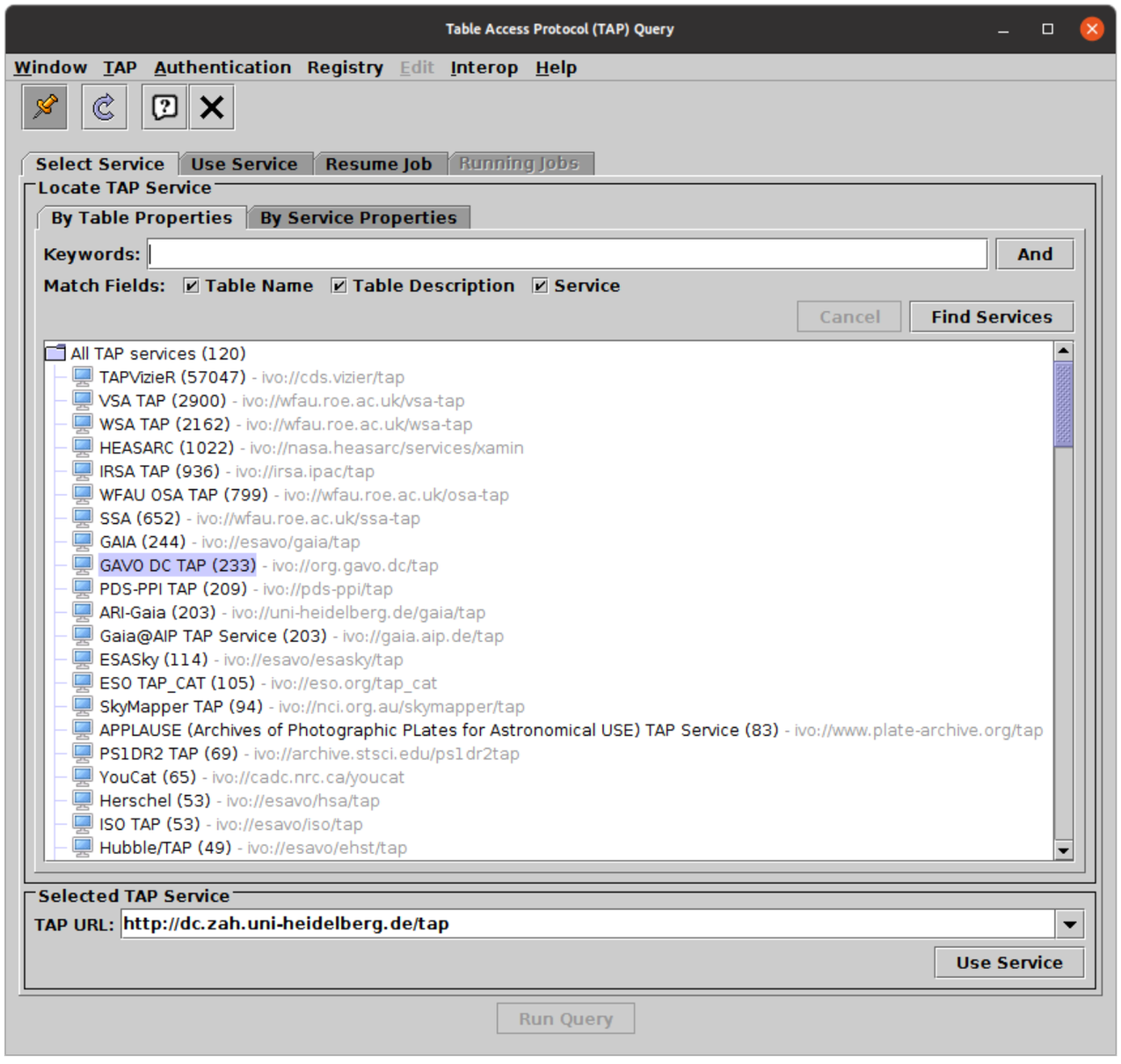




TOPCAT TAP Client

TAP window lets you

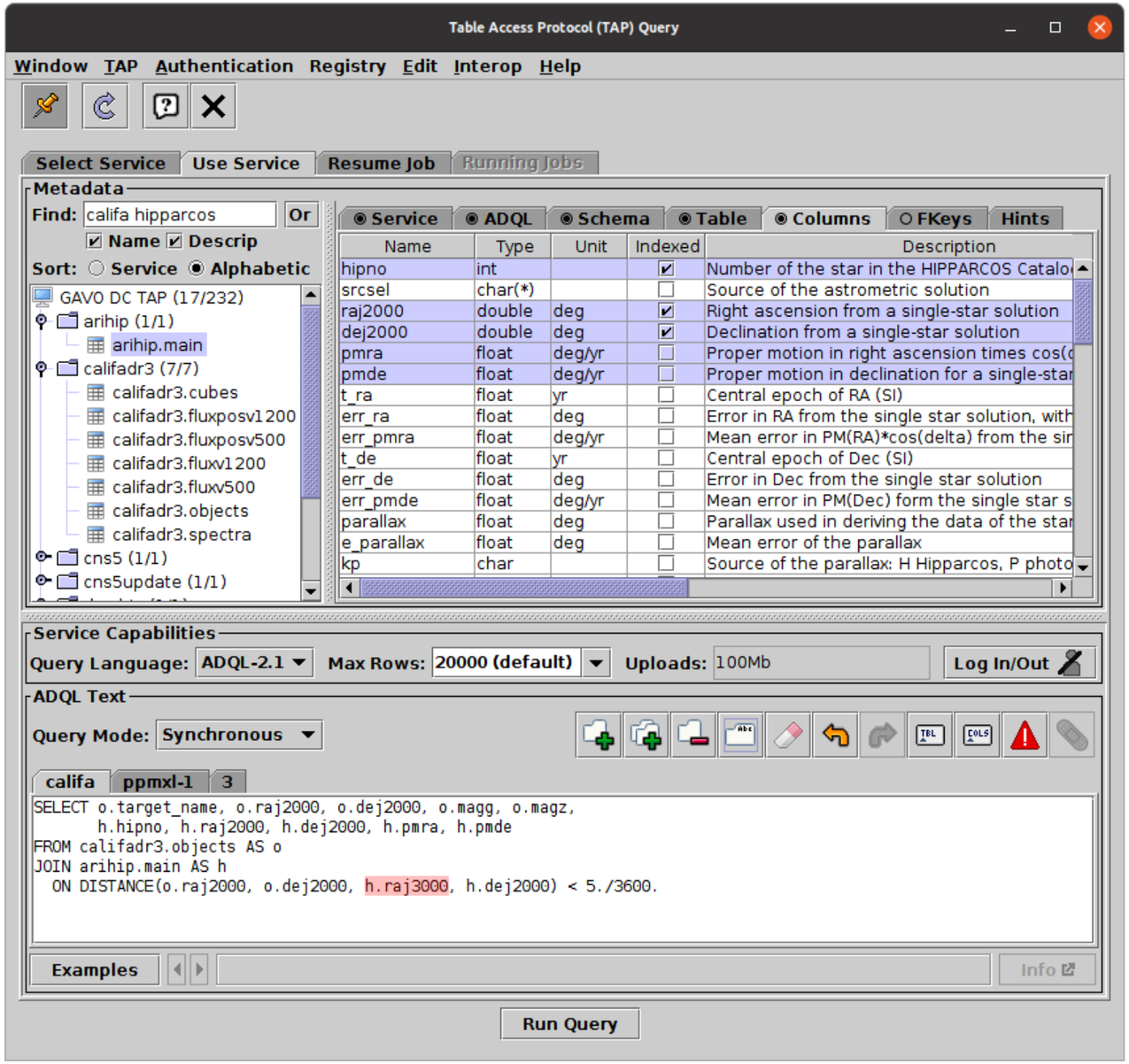
- Find what service hosts the table you want
- Browse/search tables in a service
- Browse columns in each table
- Browse standard and non-standard ADQL functions
- See column metadata (names, units, descriptions, ...)
- See/adjust service limits (max row counts etc)
- Log into restricted services (maybe)
- Enter ADQL queries
- See ADQL syntax errors highlighted
- Work from supplied Example queries
- Submit queries
- Get results (loaded into TOPCAT)



TOPCAT TAP Client

TAP window lets you

- Find what service hosts the table you want
- Browse/search tables in a service
- Browse columns in each table
- Browse standard and non-standard ADQL functions
- See column metadata (names, units, descriptions, ...)
- See/adjust service limits (max row counts etc)
- Log into restricted services (maybe)
- Enter ADQL queries
- See ADQL syntax errors highlighted
- Work from supplied Example queries
- Submit queries
- Get results (loaded into TOPCAT)



ADQL (SQL) Crash Course

SELECT Statement Structure

Roughly speaking:

```
[WITH <name> AS (SELECT ...), ...]
SELECT [TOP n] <column-or-expression> [AS <alias>] [, ...]
   FROM <table> [AS <alias>]
   [[<join-type>] JOIN <table> [AS <alias>] {USING (<columns>) | ON <condition>} [, ...]
   [WHERE <condition>]
   [GROUP BY <column-list>]
   [HAVING <condition>]
   [ORDER BY <column-list>]
```

SQL syntax is traditionally `WRITTEN IN UPPER CASE`

- but for most purposes it's all case insensitive

Example Queries (1)

Examples using GAVO DC service (Heidelberg)

- Get all table data

```
SELECT * FROM openngc.data
```

- ... up to limit imposed by server — increasing limit if required

```
SELECT * FROM hipparcos.main
```

- Limit row count (`TOP`)

```
SELECT TOP 1000 * FROM hipparcos.main
```

- Limit columns

```
SELECT TOP 1000 source_id, ra, dec, parallax FROM gaia.dr3lite
```

- Restrict by value (`WHERE`)

```
SELECT source_id, ra, dec, parallax FROM gaia.dr3lite WHERE parallax > 100
```

- Sort results (`ORDER BY`)

```
SELECT TOP 100 source_id, ra, dec, parallax FROM gaia.dr3lite WHERE parallax IS NOT NULL
ORDER BY parallax DESC
```

- ... and without an index

```
SELECT TOP 100 source_id, ra, dec, radial_velocity FROM gaia.dr3lite
ORDER BY radial_velocity
```

Example Queries (2)

Examples using Gaia TAP service (ESAC)

- Calculated columns

```
SELECT source_id, ra, dec, SQRT(pmra*pmra+pmdec*pmdec) AS pm FROM gaiadr3.gaia_source
WHERE random_index < 10000
```

- Count rows (`COUNT`)

```
SELECT COUNT(*) FROM gaiadr3.vari_rrlyrae
```

- Aggregate functions (`GROUP BY`)

```
SELECT best_classification, COUNT(*) as NUM, AVG(pf) AS pf, AVG(peak_to_peak_g) AS p2pg
FROM gaiadr3.vari_rrlyrae GROUP by best_classification
```

- Join by key (`JOIN USING`)

```
SELECT pf, peak_to_peak_g, best_classification, ra, dec FROM gaiadr3.vari_rrlyrae
JOIN gaiadr3.gaia_source USING (source_id)
```

Back to GAVO DC

- Sky position join (`JOIN ON`)

```
SELECT b.raj2000, b.dej2000, w.raj2000, w.dej2000, w1mag, w2mag, jmag, hmag, kmag
FROM browndwarfs.cat AS b
JOIN wise.main AS w ON DISTANCE(b.raj2000, b.dej2000, w.raj2000, w.dej2000) < 1./3600.
```

- Sky aggregate functions using HEALPix

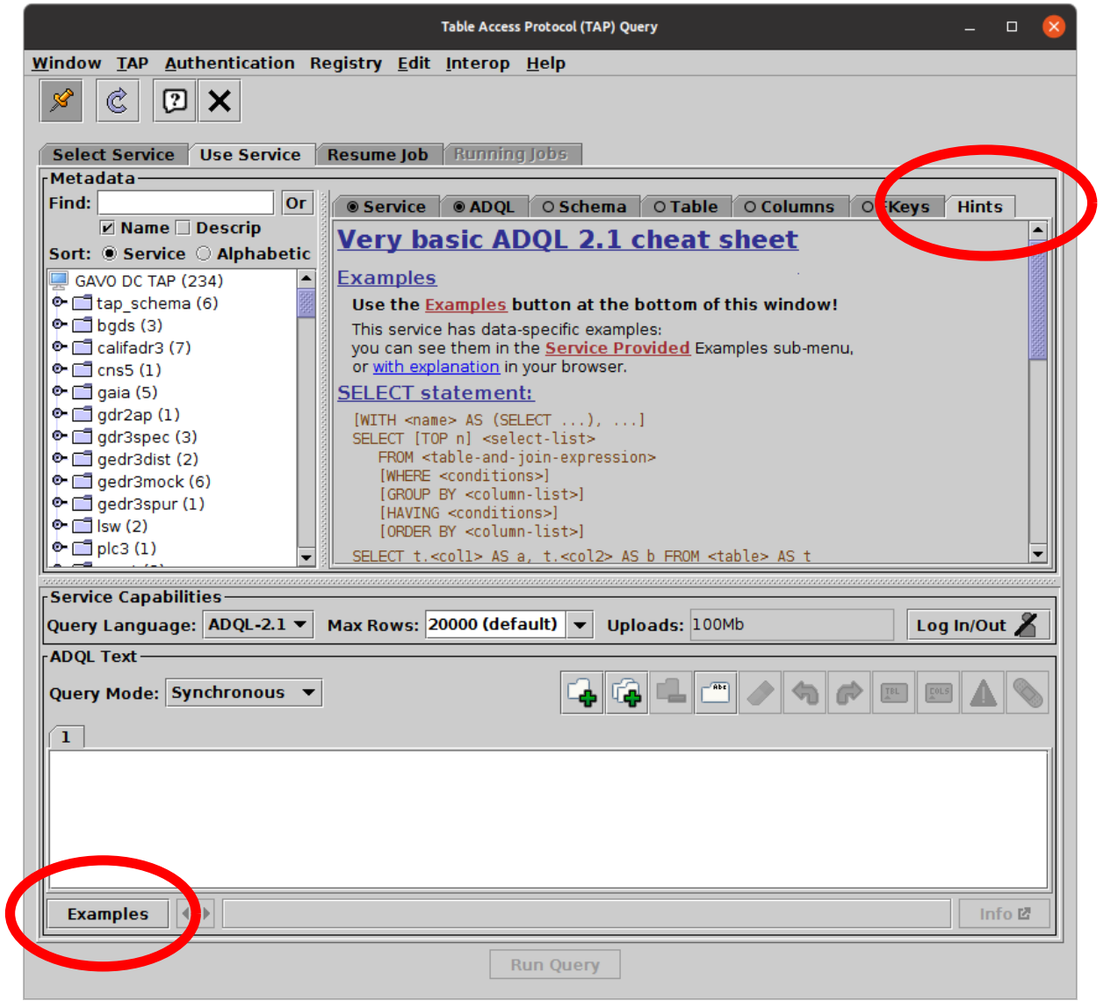
```
SELECT ivo_healpix_index(8, ra, dec) AS hpx8, COUNT(*) as num FROM apass.dr10 GROUP BY hpx8
```

More Help

Use the **Examples** menu at the bottom of the TAP window

- **Basic:** simple cribs for selection, cone, sky joins, ...
- **Service-Provided:** specific to the current TAP service

And the **Hints** tab for basic ADQL reminders



And also ...

There are plenty of TAP things I didn't mention:

- Subqueries/Common Table Expressions
- Table uploads
- Authentication
- ADQL 2.0/2.1 distinctions
- Using unregistered TAP services
- UCDs
- TAP_SCHEMA
- ObsCore
- RegTAP
- ...

TAP can do a lot!