# ObsTAP Issues for services and implementation

# ObsTAP – short names & units

- query by data model requirement: user can send the same query to all participating TAP services
  - utypes are not syntactically legal ADQL column names
  - users should not see utypes anyway
  - ➔ mandate the column names
  - ADQL syntax does not permit specifying units for numeric constants
  - ➔ mandate the units

# ObsTAP – short names & units

- simple implementation:
  - create table or view with standard table/column names and units
  - no need for ADQL parser + query processing
    - in keeping with ADQL design principles
- plausible implementation
  - use ADQL parser + query processing to map standard table/column names to internal names
  - apply unit conversion from standard to local units (admittedly this is a little tricky, but it probably helps to make best use of indices)

## ObsTAP – data access

- VOTable appendix offers this suggestion:

```
<LINK href=”http://datacenter.net/get/${obs_id}”/>
```

  - only proposed, not part of standard :-(

# ObsTAP – data access

- VOTable appendix offers this suggestion:

```
<LINK href=”http://datacenter.net/get/${obs_id}”/>
```

  - only proposed, not part of standard :-(

- a very simple solution: use info and concatenate the required obs_id, eg:

```
<INFO name=”dataAccessURL”
        value=”http://data.centre1.net/get/”/>
<INFO name=”dataAccessURL”
        value=”http://data.center2.net/get?id=”/>
```

# ObsTAP – spatial querying

- several use cases involve queries where observations contain coordinate values

```
CONTAINS(POINT('ICRS',16,41), obstap.s_bounds) = 1
```

  - databases without geometry support can still provide approximate support: use bounding box to find superset (good enough)

```
ra1 <= 16 and 16 <= ra2 and dec1 <= 41 and 41 <= dec2
```

  - have to parse ADQL and replace predicate, only service knows the coordinate range
  - services with geometry support can make use of it

# ObsTAP – UPLOAD

- several use cases involve queries with lists of input values, eg coordinates
  - ObsTAP services **must** support UPLOAD
  - if uploaded table has a column with STC-S Position and xtype=”adql:POINT”:

CONTAINS(TAP_UPLOAD.mytable.coords, s_bounds) = 1

# ObsTAP – UPLOAD

- if uploaded table has columns with RA and DEC values:

```
CONTAINS( POINT('ICRS GEOCENTER',
          tap_upload.mytable.ra, tap_upload.mytable.dec),
          s_bounds ) = 1
```

- this is not impossible for TAP service to handle correctly
  - much harder if coordinate system differs from internal and transformation is required
- how can we make this easier for everyone?
  - discourage use of separate frame+refpos,long, and lat columns