Querying using utypes

F.Bonnarel
CDS

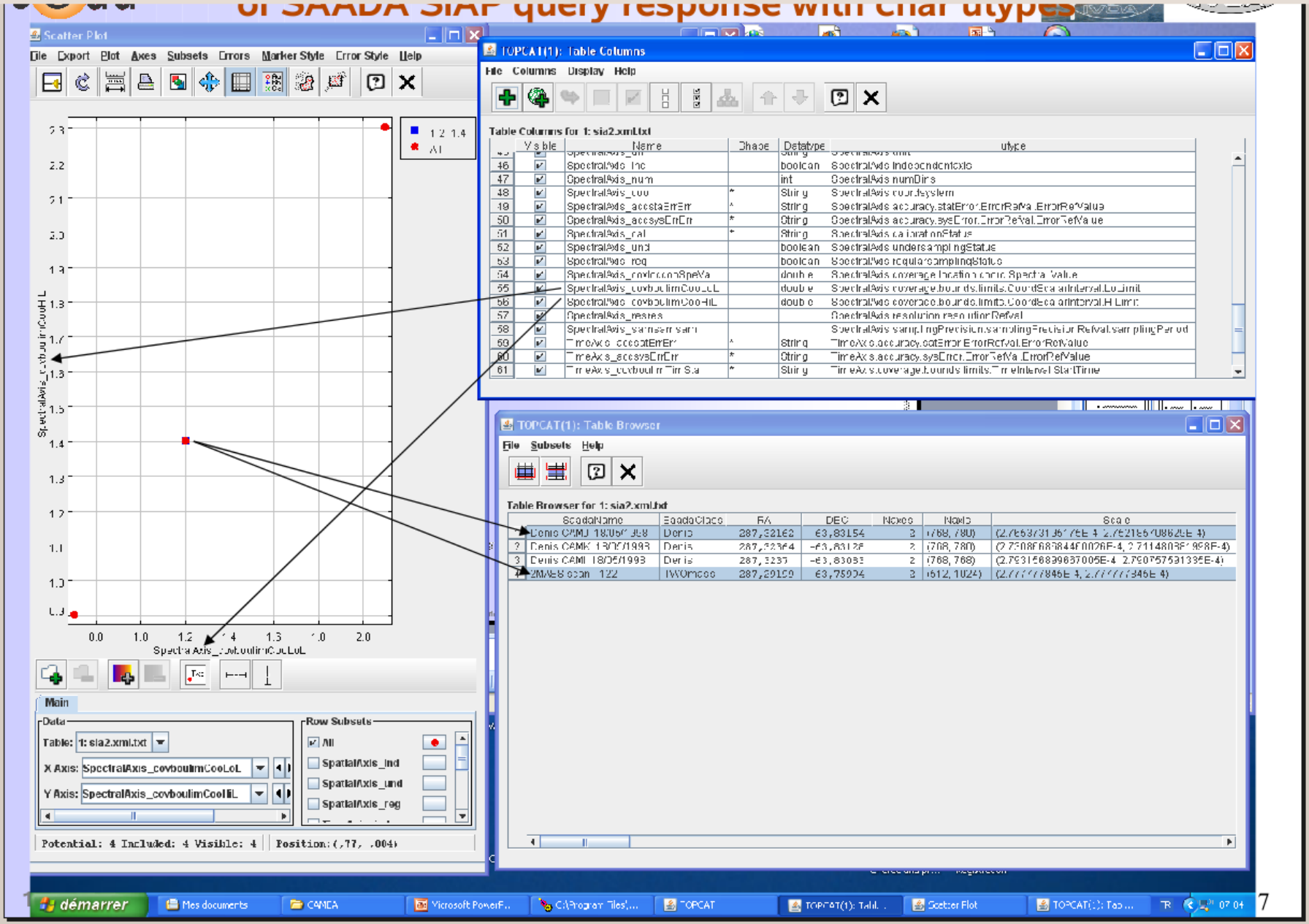
Data discovery

- Data discovery is one of the main VO Pilars
- Idea of this talk:

  --> Querying on data models is one of the most important use cases for utypes.
- Consequences on utype syntax

Browsing and selection in the Query response

- Utypes give data model mappings for attributes in the query response
- This allows to select subsets of the Query response using utypes
- Possible implementations:
  - Aladin filters
  - TOPCAT plots



Selecting by utypes using TOPCAT responses plots
(Bonnarel 2008, thanks to Mark Taylor)

Data Discovery in the VO: various data queries

- Specialized interface queries : SSA, SIA
  - (POS=, SPECRES=, etc ...
- Obstap : query on columns (small subset of Observation model) …
  - Query using ADQL
  - common relational model linked to IVOA data model
- PQL standard parameters:
  - (POS= , BAND =
- Querying on data model
  - constraint on any data model attribute

Querying on data model

- Why using query on data models?
  - Allows more flexibility than protocols
    - Can query archives missing attributes required by protocols such as ObsTAP and SimTAP
    - Full model query is possible
- Why using utypes ?
  - They have unique definition in the VO …
    - Allow interoperability between various DIFFERENT services attributes have the same definition everywhere, whatever the real structure/implementation of the archive
    - Fine tuned query
  - They get down to the lower level of the data model (Louys, 2009)
    - All ivoa attributes can be seen as « fields » in a huge Table

Querying by utypes: history

- ASPID-SR (I.Chilingarian, 2006/2007):
  - utype is an xpath in a virtual xml database
    (actually implemented as xml serialization in PostgresQL extension)
- SAADAQL (L.Michel 2008):
  - query by UCD/utypes
- Nowadays: PQL
  - PQL note (Dowler et al 2011)
    - CADC Prototyping in CAOM

1a. Find all datasets where spectral resolution is better than 80 km/s (using XPath-level constraints)

```
SELECT id,objname FROM chartest WHERE
    xpath_array(metadata,
        '//characterizationAxis[axisFrame/ucd=''em'' and resolution/resolutionRefVal/ReferenceValue < 80.0]'
        ) IS NOT NULL;
```

Results:

```
 id | objname
----+---------
  1 | IC3653
  3 | IC3509
(2 rows)
```

1b. Find all datasets where spectral resolution is better than 80 km/s (using SQL WHERE clause)

```
SELECT id,objname FROM chartest WHERE
    xpath_number(metadata,
        '//characterizationAxis[axisFrame/ucd=''em'']/resolution/resolutionRefVal/ReferenceValue'
        ) < 80.0;
```

Results:

```
 id | objname
----+---------
  1 | IC3653
  3 | IC3509
(2 rows)
```

```
2. Select equatorial coordinates as pgSphere spoint
SELECT id,objname,spoint(
      pi()*xpath_number(metadata,
        '//characterizationAxis[axisFrame/ucd=''pos'']/coverage/location/coord/Position2D/Value2/C1/text()'
      )/180.0,
      pi()*xpath_number(metadata,
        '//characterizationAxis[axisFrame/ucd=''pos'']/coverage/location/coord/Position2D/Value2/C2/text()'
      )/180.0)
    FROM
        chartest;
```

Results:

```
 id | objname |               spoint
----+---------+-------------------------------------
  1 | IC3653  | (12h 41m 29.711s , +11d 22m 01.00s)
  2 | IC3468  | (12h 32m 14.202s , +10d 15m 05.51s)
  3 | IC3509  | (12h 34m 11.503s , +12d 02m 59.39s)
(3 rows)
```

```
3. Cone search using pgSphere
SELECT id,objname
    FROM
        chartest
    WHERE
        spoint(
          pi()*xpath_number(metadata,
            '//characterizationAxis[axisFrame/ucd=''pos'']/coverage/location/coord/Position2D/Value2/C1/text()'
          )/180.0,
          pi()*xpath_number(metadata,
            '//characterizationAxis[axisFrame/ucd=''pos'']/coverage/location/coord/Position2D/Value2/C2/text()'
            )/180.0) @ '<(188.5d,11.1d),1.0d>'::scircle;
```

Results:

```
 id | objname
----+---------
  2 | IC3468
  3 | IC3509
(2 rows)
```

```
4. Query on the "time" axis
select id,objname
    FROM
        chartest
    WHERE
        ((xpath_array(metadata,
            '//characterizationAxis[axisFrame/ucd=''time'']/coverage/location/coord/text()'
         ))[1])::varchar::timestamp > '2005-01-01T00:00:00'::timestamp;
```

Results:

```
 id | objname
----+---------
  3 | IC3509
(1 row)
```

Querying by utypes: history

- ASPID-SR (I.Chilingarian, 2006/2007):
  - utype is an xpath in a virtual xml database
    (actually implemented as xml serialization in PostgresQL extension)
- SAADAQL (L.Michel 2008):
  - query by UCD/utypes
- Nowadays: PQL
  - PQL note (Dowler et al 2011)
    - CADC Prototyping in CAOM

Applications Raccourcis Système mer. 19 oct., 05:44 bonnarel

Slide 1

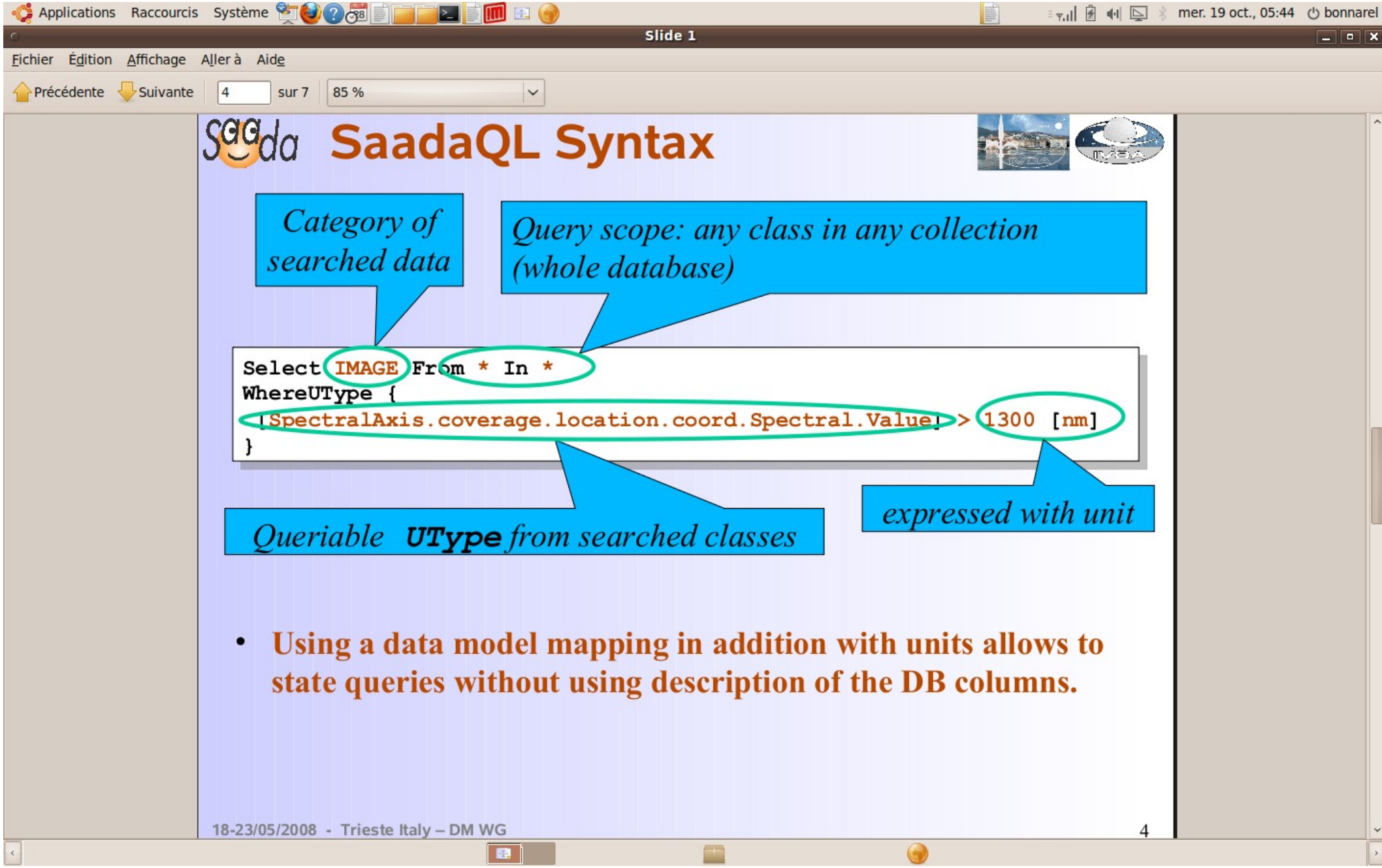
Fichier Édition Affichage Aller à Aide

Précédente Suivante 3 sur 7 85 %

Steps for a Solution

- **Building a SaadaDB**
  - Create an empty SaadaDB with Saada
  - Create a Saada collection in this SaadaDB (e.g. **`VoDMinPractice`**)
  - Load successively both image sets
- **Tag DB Image Columns with Characterization `UTypes`**
  - By hand with the SaadaDB admin tool (to show how to proceed)
  - With a script (to save demo time)
- **Select Images in the Merged Collection**
  - Using SaadaQL queries based on **`UTypes`**
  - Using the SaadaDB SIAP server with *Aladin*

Querying by utypes in SAADAQL . L.MIchel 2008



Querying by utypes in SAADAQL . L.MIchel 2008

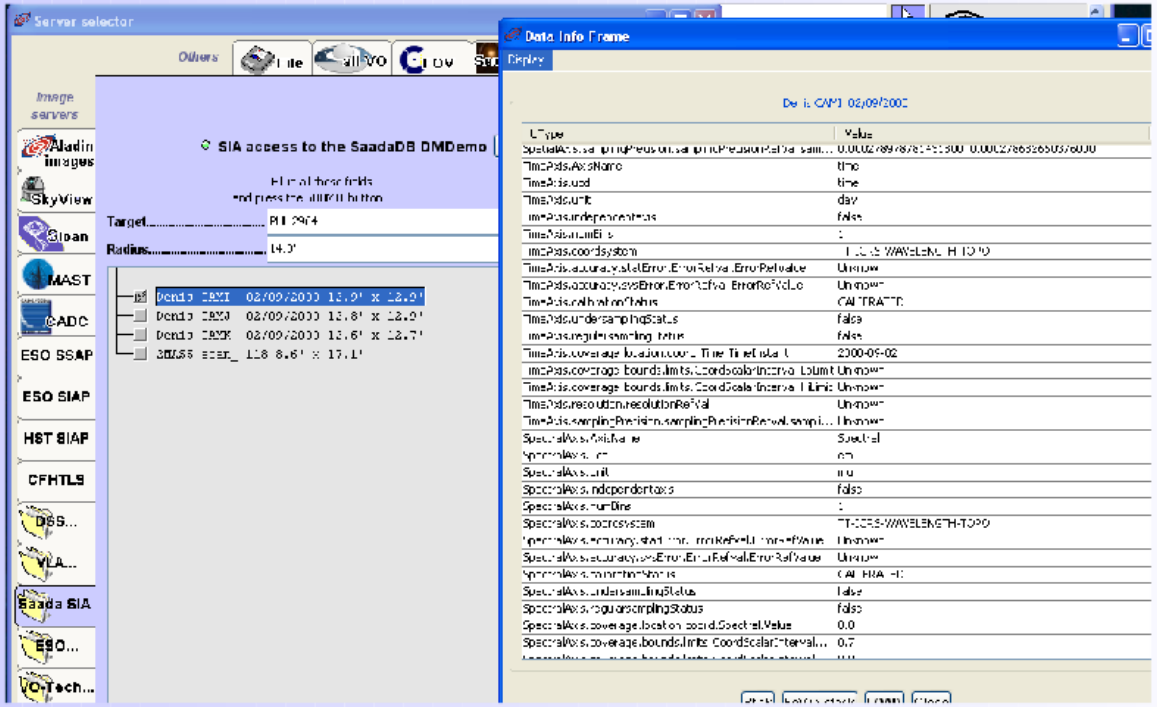
Applications Raccourcis Système mer. 19 oct., 05:44 bonnarel

Slide 1

Fichier Édition Affichage Aller à Aide

Précédente Suivante 5 sur 7 85 %

Aladin display of SAADA SIAP query response with char utype



Querying by utypes in SAADAQL . L.MIchel 2008

Querying by utypes: history

- ASPID-SR (I.Chilingarian, 2006/2007):
  - utype is an xpath in a virtual xml database
    (actually implemented as xml serialization in PostgresQL extension)
- SAADAQL (L.Michel, 2008):
  - query by UCD/utypes
- Nowadays: PQL
  - PQL note (Dowler et al, 2011)
    - CADC Prototyping in CAOM

Two PQL-utype query examples (from CADC)

PQL query using the ivoa.ObsCore data model utypes:

```
REQUEST=doQuery
LANG=PQL
obscore:DataID.collection=JCMT
obscore:Obs.dataProductType=spectrum
```

RSS feed of calibrated optical images from CAOM -- using utypes only!:

```
REQUEST=doQuery
FORMAT=rss
LANG=PQL
caom:Plane.calibrationLevel=2
caom:Plane.energy.emBand=Optical
caom:Plane.dataProductType=image
```

PQL utype queries
CADC

- Assumes querying a TAP service
- Column / utype mapping and joins stored in the TAP_SCHEMA
- Applications:
  - RSS (or non VOTABLE) formats
  - Non Obscore archives with IVOA Utypes
  - Non IVOA (proprietary) models

Constraints on utypes syntax

- No combination of utypes = single string
  - Otherwise ambiguous
- Namespace is not changing the meaning, only gives the general context:
  - sia:DataID.ObservationID

  is the same as

  obs:DataID.ObservationID
  - This is a requirement for interoperability

Conclusions

- Current status of the note need few modifications needed

- We need a syntax recommendation soon to support queries by utypes,

  a major use case

- Thanks