ADQL/s Syntax (Proposal)
- towards unification of ADQL, SIAP, SSAP, SXAP... -

Yuji SHIRASAKI
yuji.shirasaki@nao.ac.jp
National Astronomical Observatory of Japan
JVO

Objective of this talk

- Establish a unified query language used to search for all kinds of Astronomical Data, such as, Catalog, Image, Spectrum, 3D-Data Cube, Photon List, Light Curve, ...
- Current situation
  - Catalog Data Search — ADQL
  - Image Data Search/Retrieval — SIAP
  - Spectrum Data Search/Retrieval — SSAP
  - others — nothing
- We should define a master language whose semantics is upward compatible with SIAP and SSAP ...
- I present proposed extensions of SQL syntax (JVOQL V.2) for Astronomical Data Query Language (ADQL/s).

What is Lacking in SQL

- the way to describe a point and/or a region in Space, Spectrum and Time coordinate.
- the way to describe a catalog cross match condition.
- the way to search observational data (Image, Spectrum, Data Cube, Photon List, ...)

Extension of standard SQL

- UCD support in Select list (Skip)
- Table name Identifier in FROM clause (Skip)
- Extension on data type
- Operators for the extended data types
- Functions for the extended data types

3D Geometry data type of PostgreSQL

We can learn from PostgreSQL

- Geometry data type is defined in PostgreSQL.
- Geometry data type has the following sub data type:

  Point (x,y), Box ((x1,y1),(x2,y2)),

  Polygon ((x1,y1),...), Circle <(x,y),r>, etc...
- Geometry data operators

  circle '((0,0),2)' ~ point '(1,1)' includes

  polygon '((0,0),(1,1))' ~= polygon '((1,1),(0,0))' equals
- Geometry data function

  circle(point, double precision) returns circle data type

  box(point, point) returns box data type

Extension of Data types (minimum req.)

Data Type Name	Definition	Example
<Sky> :=	<SkyCoord> | <SkyPoint> | <SkyROI>	
<SkyCoord> :=	<SkyCoordAngle> | <SkyCoordSexag>	
<SkyCoordAngle> :=	NUMBER [deg|arcmin|arcsec|…]	30 deg, 15 arcmin
<SkyCoordSexag> :=	hh:mm:ss.sss | [+|-]dd:mm:ss.sss	270:30:20.20, -30:00:12.3
<SkyPoint> :=	([<CoordSys>], <SkyCoord>, <SkyCoord>)	('FK5', 200. deg, +30 deg)
<CoordSys> :=	'[B1950|J2000] (ICRS|FK4|FK5) [Equ|Gala|…]'	'ICRS'
<SkyROI> :=	<SkyCircle> | <SkyBox> | <SkyPolygon>	
<SkyCircle> :=	(<SkyPoint>, r [unit])	(('FK5',200,+30), 1.0 arcsec)
<SkyBox> :=	(<SkyPoint>, <SkyPoint>)	(('FK5',200,+30), (210,+32))
<SkyPolygon> :=	(<SkyPoint>, <SkyPoint>, <SkyPoint>[, …])	((200,20),(200,30),(210,30),(210,20))
<Spectrum> :=	<SpectrumCoord> | <SpectrumROI>	
<SpectrumCoord> :=	NUMBER (A|nm|um|mm|cm|Hz|eV|…)	500 nm, 6 keV
<SpectrumROI> :=	<SpectrumRange> | <SpectrumBand>	
<SpectrumRange> :=	['('|'['] <SpectrumCoord> .. <SpectrumCoord> [')'|']'	100nm..500nm, [1 keV..1000 nm)
<SepctrumBand> :=	'STRING'	'Xray', 'Optical', 'B', 'V'
<DateTime> :=	'yyyy-yy-yy [hh:mm:ss.ss]' | MJD	
<DateTimeROI> :=	(<DateTime> .. <DateTime>)	
<ObsData> :=	<DataCube> | <Image> | <Spectrum> | <Photon> | <LightCurve> | …	
<DataCube> :=	<DataCubeEntity> | <DataCubePointer>	
<Image> :=	<ImageEntity> | <ImagePointer>	
<Spectrum> :=	<SpectrumEntity> | <SpectrumPointer>	
<Photon> :=	<PhotonEntity> | <PhotonPointer>	
<LightCurve> :=	<LightCurveEntity> | <LightCurvePointer>	
<XXXEntity> :=	FITS file | Jpeg | …	
<XXXPointer> :=	URL | SIAP | SSAP | ID …	

Extension of Operators

Operator	Definition	Meaning
~	<SkyROI> ~ <SkyPoint> | <SkyROI> ~ <SkyRegion> | <SpectrumROI> ~ <SpectrumCoord> | <SpectrumROI> ~ <SpectrumROI> | <DateTimeROI> ~ <DataTime> | <DateTimeROI> ~ <DateTimeROI>	Left includes Right
&&	<SkyROI> && <SkyROI> | <SpectrumROI> && <SpectrumROI> | <DateTimeROI> && <DateTimeROI>	Overlap
=	<SkyPoint> = <SkyPoint> | <SkyROI> = <SkyROI> | <SpectrumCoord> = <SpectrumCoord> | <SpectrumROI> = <SpectrumROI> | <DateTime> = <DateTime> | <DateTimeROI> = <DateTimeROI>	Left equals Right (extension for '=' to be used for new data types.)
BETWEEN ... AND ...	<SpectrumROI> between <SpectrumCoord> and <SpectrumCoord> | (<DateTime>|<DateTimeROI> between <DateTime> and <DateTime>	(extension for 'BETWEEN' to be used for new data types.)

Functions

Minimum Requirement for simple cross match.

SkyCoordAngle Distance(SkyPoint p1, SkyPoint p2)

Optional.

SkyPoint SkyPoint(SkyCoordAngle x, SkyCoordAngle y)

SkyROI SkyROI(String regionDescription)

SkyROICircle SkyROI(SkyPoint p, SkyCoordAngle r)

SkyROIBox SkyROI(SkyPoint p, SkyCoordAngle w, SkyCoordAngle h[, SkyCoordAngle PA])

SkyROIPolygon SkyROI(SkyPoint[] p)

SpectrumROI SpectrumROI(SpetrumCoord s1, SpectrumCoord s2)

SpectrumROI SpectrumROI(SpectrumBand bandName)

DateTimeROI DateTimeROI(DateTime d1, DateTime d2)

ImagePointer[] ImagePointer(SkyROI region[, SpectrumROI spe][, DateTimeROI d])

SpectrumPointer[] SpectrumPointer(SkyROI region[, SpectrumROI spe][, DateTimeROI d])

PhotonPointer[] PhotonPointer(SkyROI region[, SpectrumROI spe][, DateTimeROI d])

Expressions to specify a point and a region in Space Coordinate

A point in the Sky Coordinate ("SkyPoint" data type):

(180, 30) -- Default coordinate frame of each data service is applied.

('Gala J2000', 180, 30) –- Galactic coordinate (J2000) in degree

('ICRS', 12:00:00.0, +30:00:00.0) –- Equatorial in Sexagecimal

A region in the Sky Coordinate ("SkyROI" data type):

((180, 30), 1.0 deg) -- A circle centered at (180, 30) and 1 deg radius.

((180, 30), (181, 31)) -- A rectangle whose coordinate of opposite points are at (180, 30) and (181,31).

(('ICRS',10,20), (12,20), (12,22), (10,22), (10,20)) –- A polygon. Coordinate frame specified in the first point is inherited by the following points.

"SkyPoint" and "SkyROI" are sub data types of "Sky" data type

Expressions to specify a region in Spectrum and Time Coordinate

A SpectrumROI data type (“SpectrumRange” data type):

(100 nm .. 800 nm) –- use “..” operator (like a perl script).

[10 keV .. 300 GHz) –- includes 10 keV, excludes 300 GHz.

100 .. 800 –- equivalent to [100..800), default unit of each data service is applied.

Another SpectrumROI type (“SpectrumBand” data type)

“V”, “B”, “Soft-X”, “Hard-X” –- any string describing a spectrum band. The data server should define spectrum range for each world.

A time region (“DateTimeROI” data type):

(‘2004-03-20’ .. ‘2004-05-20’) –-

(‘2004-05-20 10:30:00.00’ .. ‘12:00:00.0’) --

Cross match

IVOA SkyNode Interface Version 0.7.4 proposes the following construct for cross matching:

```
Where Xmatch(table1, table2, !table3) < presicion
```

My thought, however, is that it is not appropriate to put the Xmatch function(?), which does not take any arguments on column, in “WHERE” clause, since “WHERE” clause is the place where selection criteria for each select record is described.

So, I propose to add the following “XMATCH” clause to select command or to use distance() function for a simple cross matching purpose:

```
Xmatch (table1.(ra, dec), table2, !table3) < presicion1
                                and (table1, table4) < precision2
```

Examples of Usage

Select objects in a specified sky region

Select objects which are located inside a circle centered at RA=19.5deg Dec=-36.7 (ICRS) with radius 0.02 degree.

```
select  a.*  from  Tab a
where ((‘ICRS’, 19.5, –36.7’), 0.02 deg) ~ (a.ra, a.dec)
```

The following query is from ADQL WD Version 0.7.4.

“Region” should be compared with “point”, but omitted....

```
select  a.*  from  Tab a
where  Region(‘Circle J2000 19.5 –36.7 0.02’)
```

It is obvious which coordinate to be compared if “Tab” has only one set of coordinate columns, but it is not necessary the case, e.g. a cross-idenfitied table might have two sets of coordinate columns. I propose to explicitly specify which coordinate columns is tested.

Cross Match

User Distance() function or Xmatch Clause.

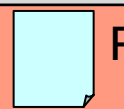
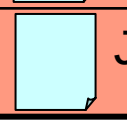
```
select   a.ra, a.dec, a.mag, b.ra, b.dec, b.flux
from     OpticalTable a, XrayTable b
where  (('ICRS', 19.5, –36.7), 0.02 deg) ~ (a.ra, a.dec) and
         Distance((a.ra, a.dec), (b.ra, b.dec)) < 5 arcsec
```

```
select   a.ra, a.dec, a.mag,
          b.ra, b.dec, b.flux,  c.ra, c.dec, c.flux
from     OpticalTable a, XrayTable b, RadioTable c
where    (('ICRS', 19.5, –36.7), 0.02 deg) ~ (a.ra, a.dec)
Xmatch (a, b.(ra, dec), !c) < 5.0 arcsec
```

How is the Image Query Described in SQL ?

- SQL is a language for relational data base.
- Assume a virtual data table which has images for all possible location, size, shape, spectrum band, etc...
- The columns of the virtual data table can be categorized as “search condition column” and “data column”.

← search condition columns → ← data columns →

Region	SpectrumBand	Format	ImagePointer (SIAP?)	ImageEntity
((10,+20),0.1)	‘B’ (= 400nm .. 500nm)	‘FITS’	http://xxx/getImage?POS=...?...	 FITS
((10,+30),0.2)	‘R’ (= 600nm .. 700nm)	‘JPEG’	http://xxx/getImage?POS=...?...	 JPEG

```
select SpectrumBand, ImageEntity from optImage
where Region = ((‘ICRS’, 10, +20), 0.1) and Format=‘FITS’
      and (100nm .. 1000nm) ~ SpectrumBand
```

Procedure for Data Query

1. A Client requests the SkyNode (or Local Registry ?) column metadata for an interested table.

2. The SkyNode (or LR) return a list of “column name”, “data type”, “Unit”, “Associated Coordinate System” if any and UCD for each column.

3. The “column name” is used for keyword of data search. The “data type” or “UCD” is used to know supported region search, such as sky, spectrum and time region.

4. The “data type” and/or “UCD” provides types of the data (Image, Spectrum, Data Cube, Photon List, ...) searchable with this table.

Image Query for Multiple Regions

```
SELECT regionSky region, spectrumBand band, pointer
FROM   virtualImage
WHERE regionSky IN (
      (('ICRS', 19.5, –36.7), 0.02 deg), ((23.5,-13.0), 0.02),
      ((223.5,+23.0), 0.05) , ((123.5,+43.0), 0.07) )
   and (100 nm .. 1000 nm) ~ spectrumBand
   and obsDate = '2002-03-23' .. '2004-05-18'
```

If you know default coordinate system and unit used in this table, you can omit 'ICRS' and 'deg'

Query Result (VOTable)

region	band	pointer
((19.5,-36.7), 0.02)	'B'	SIAP description
((19.5,-36.7), 0.02)	'V'	SIAP description
((19.5,-36.7), 0.02)	'R'	SIAP description
((23.5,-13.0), 0.02)	'R'	SIAP description
((223.5,+23.0), 0.05)	'R'	SIAP description
((123.5,+43.0), 0.07)	'R'	SIAP description

If you know band name stored in this table, you can explicitly specify as "spectrumBand = 'R'"

Note that "expression IN (list)" is a standard SQL syntax.

Join between Catalog and Image Data Table

Select objects located in a box region centered at ra=270 and dec=-1.5 with 1 deg x 1 deg size and get their FITS image

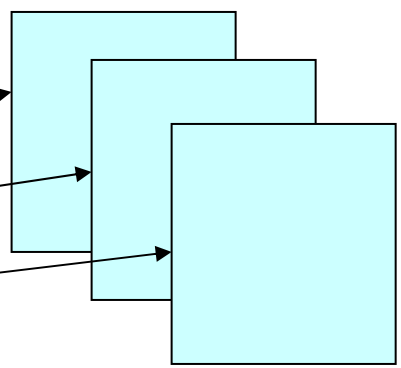
```
select  cat.ra, cat.dec, img.ImageEntity
from    ObjectCatalog cat, Image img
where   img.regionSky = ((cat.ra, cat.dec), 1.0 deg)
        and img.OutputFormat = "FITS"
        and (('ICRS', 270.0, -1.5, 0.2 deg) ~ (cat.ra, cat.dec)
```

table join condition

Region selection for catalog data

Query Result (VOTable)

cat.ra	cat.dec	img.ImageEntity
271.0	-1.50	FITS filename
271.2	-1.48	FITS filename
270.0	-1.51	FITS filename



Cross Match and Image Request

1. Select objects located in a circle region centered at ra=270 and dec=-1.5 with 0.2 deg radius from optical catalog.
2. Identify X-ray counter part for each selected object with 5 arcsec precision.
3. Get FITS images of 10” radius size from optical and X-ray image data service.

```
select    optCat.ra, optCat.dec, xCat.ra, xCat.dec,
          optImage.Image, xImage.Image
from      optCat, xCat, optImage, xImage
where     (('ICRS', 270.0 deg, -1.5 deg), 0.2 deg) ~ (optCat.ra, optCat.dec)
and Distance((optCat.ra, optCat.dec), (xCat.ra, xCat.dec)) < 5 arcsec
and optImage.regionSky = ((optCat.ra, optCat.dec), 10 arcsec)
and xImage.regionSky = ((optCat.ra, optCat.dec), 10 arcsec)
and optImage.FORMAT = “FITS”
and xImage.FORMAT = “FITS”
```

Query to a Spectrum Database by an Object Name

```
select ObjectName, Spectrum
from OpticalSpectrumTable
where ObjectName IN ('Crab Nebula', 'Vega', '3C273')
and Exposure > 300 sec and Format = "FITS"
and spectrumRange between 100 nm and 1000 nm
```

Query Result (VOTable)

spe.ObjectName	spe.ImageEntity
Crab Nubula	FITS name
Vega	FITS name
3C279	FITS name

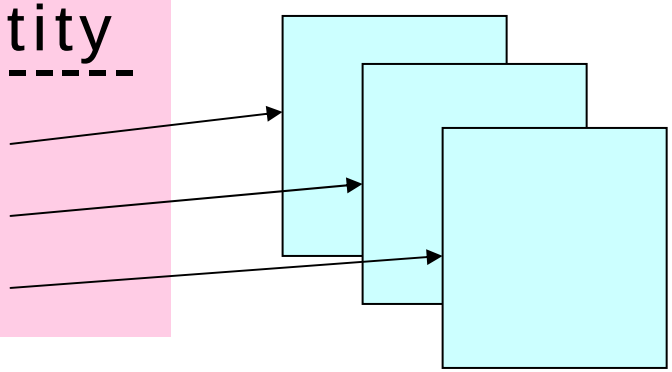


Image and Spectrum from Data Cube

```
Select ImagePointer
from ALMADataCube
where SkyRegion = (('ICSR', 12:23:12.3, +21:30:43), 0.1 deg)
and SpectrumRegion = (100 GHz .. 800 GHz)
```

"WHERE" clause specifies a sub data cube and "ImagePointer", which has an "Image" data type, describe the way of projection.

```
Select SpectrumPointer
from ALMADataCube
where SkyRegion = (('ICSR', 12:23:12.3, +21:30:43), 0.01 deg)
and SpectrumRegion = (100 GHz .. 120 GHz)
```

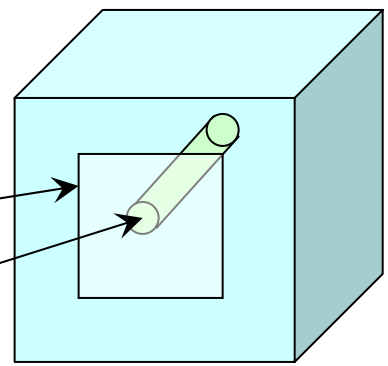
"SpectrumPointer", which has an "Spectrum" data type, describe the way of projection.

Query Result
ImagePointer

SIAP

Query Result
SpectrumPointer

SSAP



Same for a photon list database

Summary

Syntax of ADQL/s is proposed on the basis of SQL extension.

- New data types (SkyPoint, SkyROI, SpectrumROI ...) are introduced to describe a point and a region in the Space, Spectrum and Time coordinate, and their expressions are defined. point -> ("CoordSys", x, y), circle -> ((x,y), r) ...
- Boolean operators "~" (=includes) and "&&" (=overlap) are introduced to describe the region conditions.
- "XMATCH" clause is introduced to describe the cross match conditions.

XMatch (table1(ra, dec), table2, !table3) < precision