Common Interface for Every Kind of Astronomical Data Service

Under Preparation !!

Yuji Shirasaki

National Astronomical Observatory of Japan

JVO

VOQL+DAL Joint session

JVO Skynode Implementation Experience

Under Preparation

Yuji SHIRASAKI

National Astronomical Observatory of Japan/JVO

yuji.shirasaki @ jvo.nao.ac.jp

VOQL Joint session

Contents

- Objective of this talk
- Concept for adapting SkyNode Interface on DAL service
- JVO SkyNode experimental implemenation
  - SkyNode Toolkit
  - Application to the science use case
- Required Specification
  - Data Service Classification and Table Data model
  - Definition of minimum subset of ADQL
  - Minor update on ADQL
  - Standard on usage of VOTable
  - Access to the Object data type
  - Metadata access interface
  - Standardization of error message

Contents

- Introdution of JVO SkyNode toolkit
  - used free software:
  - archetectur of JVO skynode
- Implmentation problems
  - XML → Java deserialization problem in AXIS
  - Namespace problem of ADQL → JVO (v0.8) vs NVO (v0.74)
  - Namespace problem of VOTable → JVO (v1.1) vs NVO (v1.0 or no namespace)
  - Usage of VOTable → id, name attributes ...
  - Complexity of STC object
  - ...
- Proposal
  - Simplify the ADQL and STC → Define minimum subset of ADQL and STC and freeze them (never update, never change the namespace)
  - VOTable transfer → attachment or URL
  - Standardize the error message

Objective of this talk

- From a Simple Protocol to an Intelligent Protocol
  - SIAP and SSAP are simple parameter-based query protocol.
  - ADQL is a SQL-based query protocol and have a capability to describe more complicated query conditions.
- What is the merit ?
  - Multiple regions condition, directly or by VOTable
  - Homogeneous data access to both of the catalog and observational data
  - Query to the multiple data services (Portal)
- Effect on the current SIAP and SSAP spec.
  - Introducing the ADQL to the DAL service does not affect to the data model of a returned VOTable.
  - Just introduce an additional query interface to the DAL service.

Concept for adapting SQL to Image Database.

I have shown that image query can be described in SQL syntax by introducing virtual column concept.

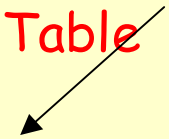
SIAP Query

POS	[mandatory]
SIZE	[mandatory]
FORMAT	[mandatory]
INTERSECT	[option]
...	

```
http://jvo.nao.ac.jp/Image?
Pos=34.3,-5.11&Size=0.01&Format=VOTable&
INTERSECT=OVERLAPS
```

SIAP parameters and returned metadata are taken as columns of the virtual table.

Image Query on Virtual Table



Pos	Size	Format	ImageURL
(23,+30)	1.0	fits	http://jvo.nao.ac.jp/Image?id=124214
(23,+23)	0.3	jpeg	http://jvo.nao.ac.jp/Image?id=124215
...	...	...	...

```
Select  imageURL
From    imageTable
Where   Pos = Point(23,+30)
        and Size = 1.0 and
        Format = 'fits'
```

This is a virtual table which has infinite number records.

Development of the JVO SkyNode Toolkit

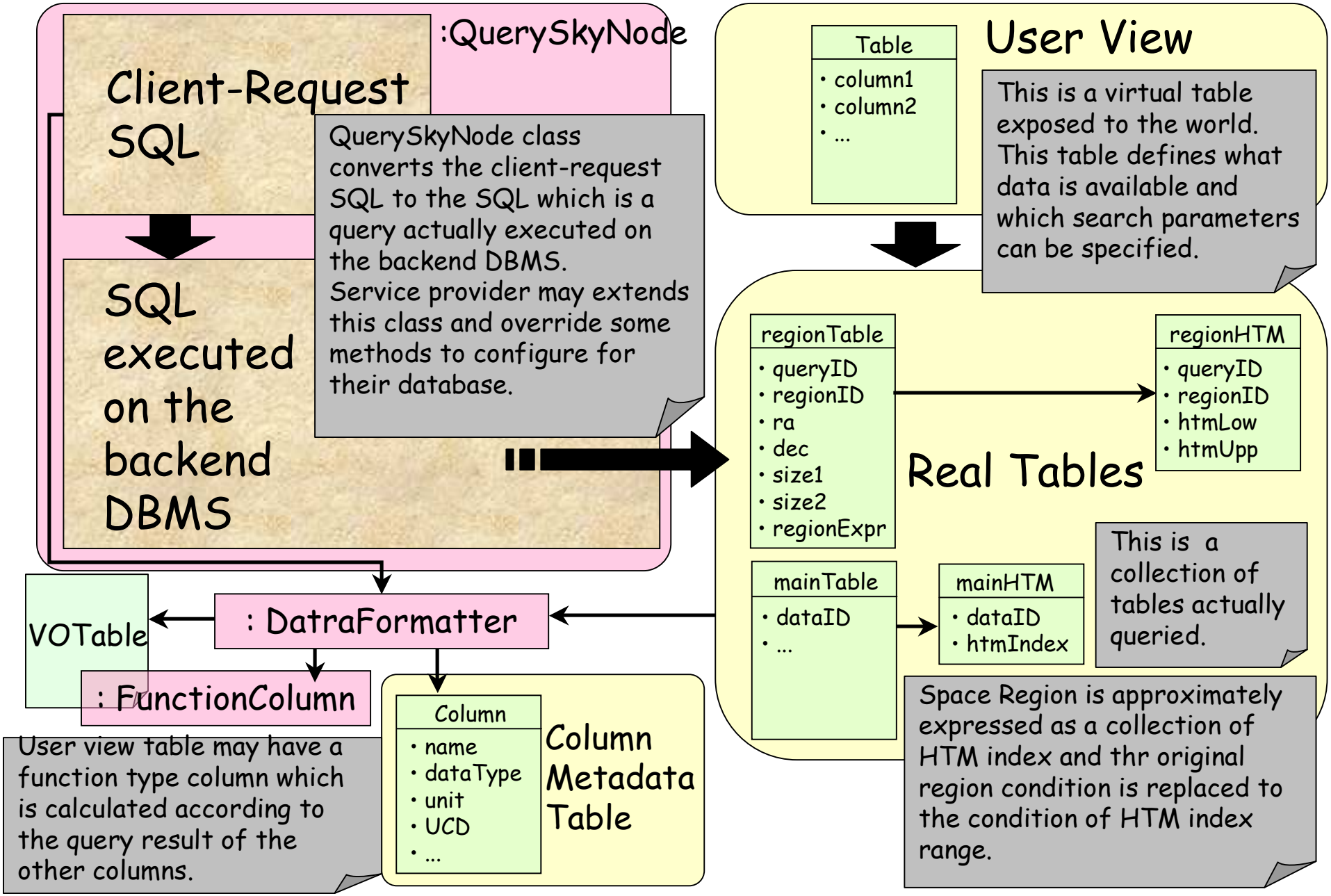
- Primary aim: to provide a reference implementation for a DAL service using SkyNode/ADQL/VOTable interface
- Independent from the type of backend DBMS (PostgreSQL, Oracle, MySQL, …) JDBC driver is required.
- Restrictions:
  - Not all the ADQL syntax are supported.
  - String representation of ADQL is JVOQL.
- Experimental Release:
  - http://jvo.nao.ac.jp/download/skynode-toolkit

Software used

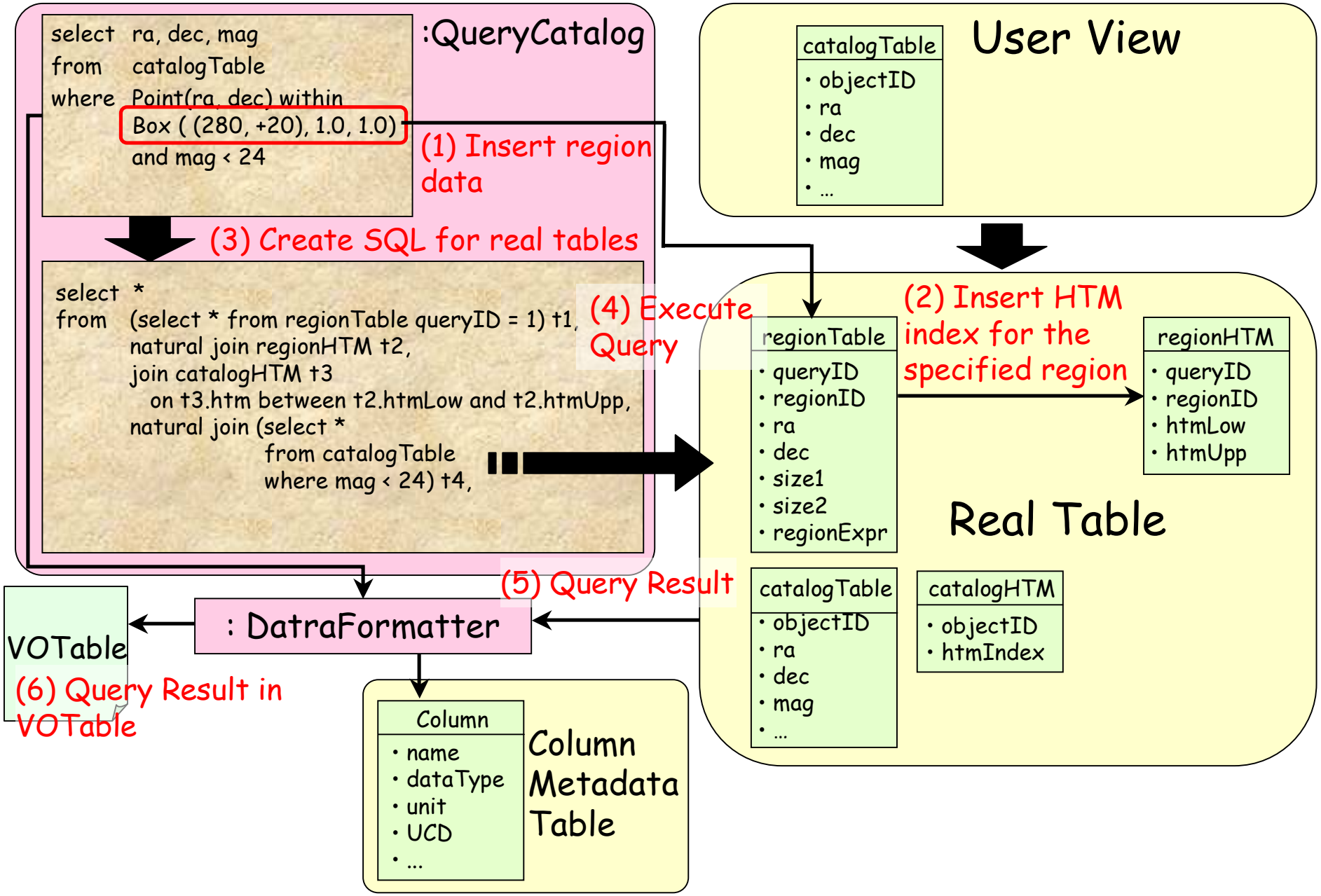
- Tomcat 4.1.31 ---- servlet container
- Axis 1.2RC1 ---- web service engine
- J2SDK 1.4.2 ---- Java compiler & libary
- Ant 1.6.1 ---- Java-based build tool
- JavaCC 3.2 ---- parser generator for Java
- PostgreSQL 7.4.7 ---- DBMS
- Java HTM library (JHU) ---- spherical indexing
- Java FITS library (HEASARC) ---- FITS IO lib
- ...

Architecture

JVO SkyNode Toolkit Flow Chart



Catalog Data Query by ADQL



Catalog Data Xmatch Query with VOTable

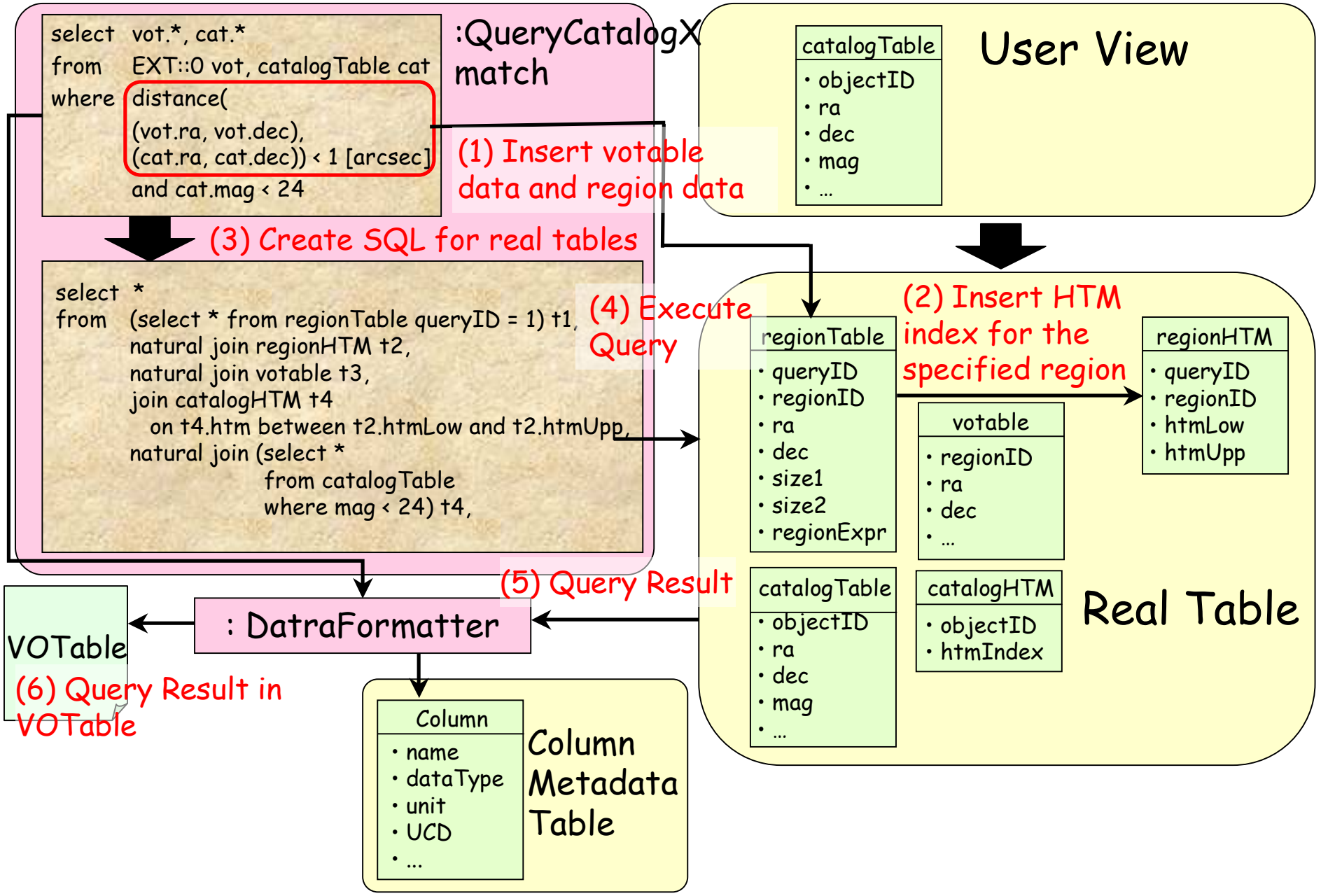


Image Data Query by ADQL

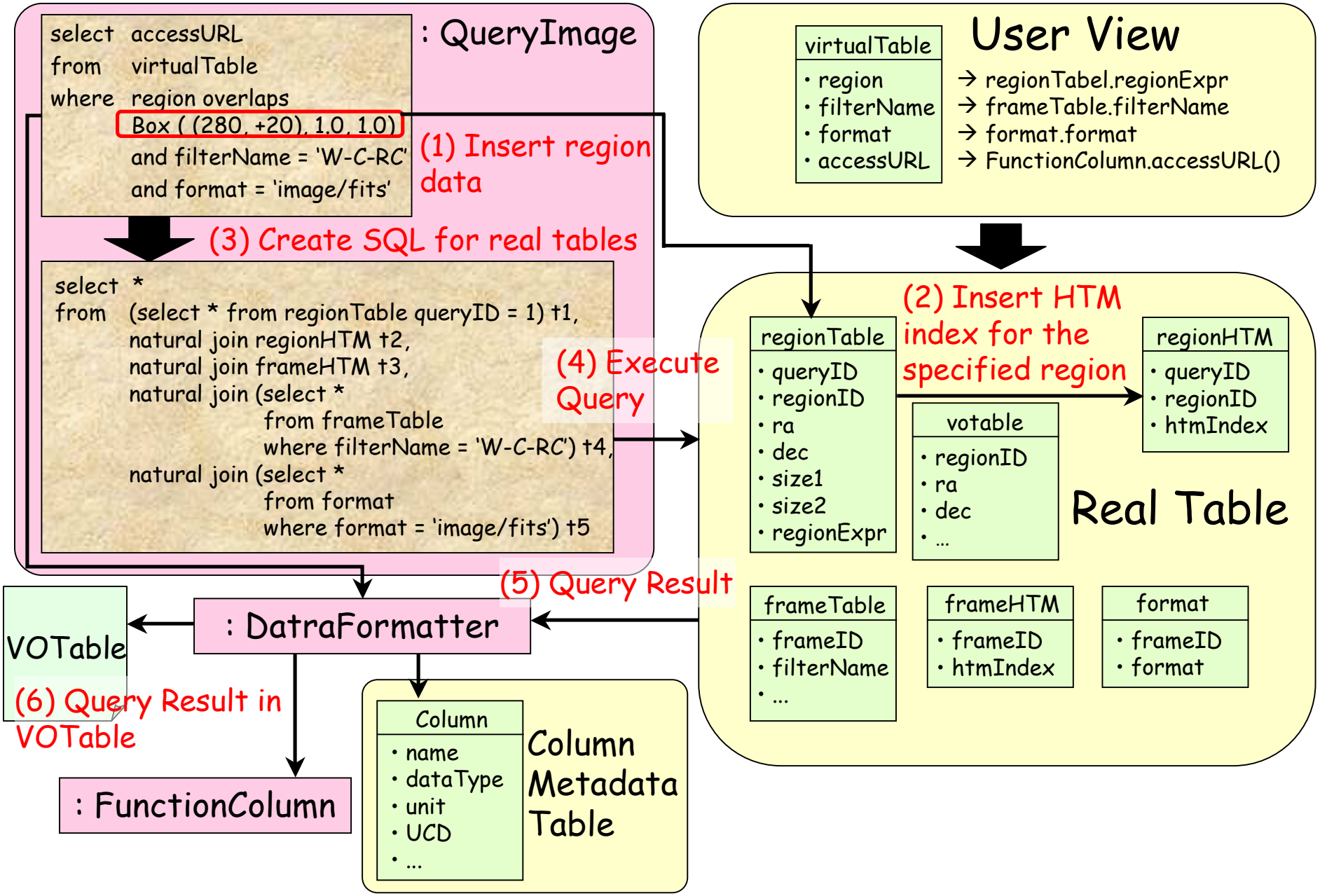
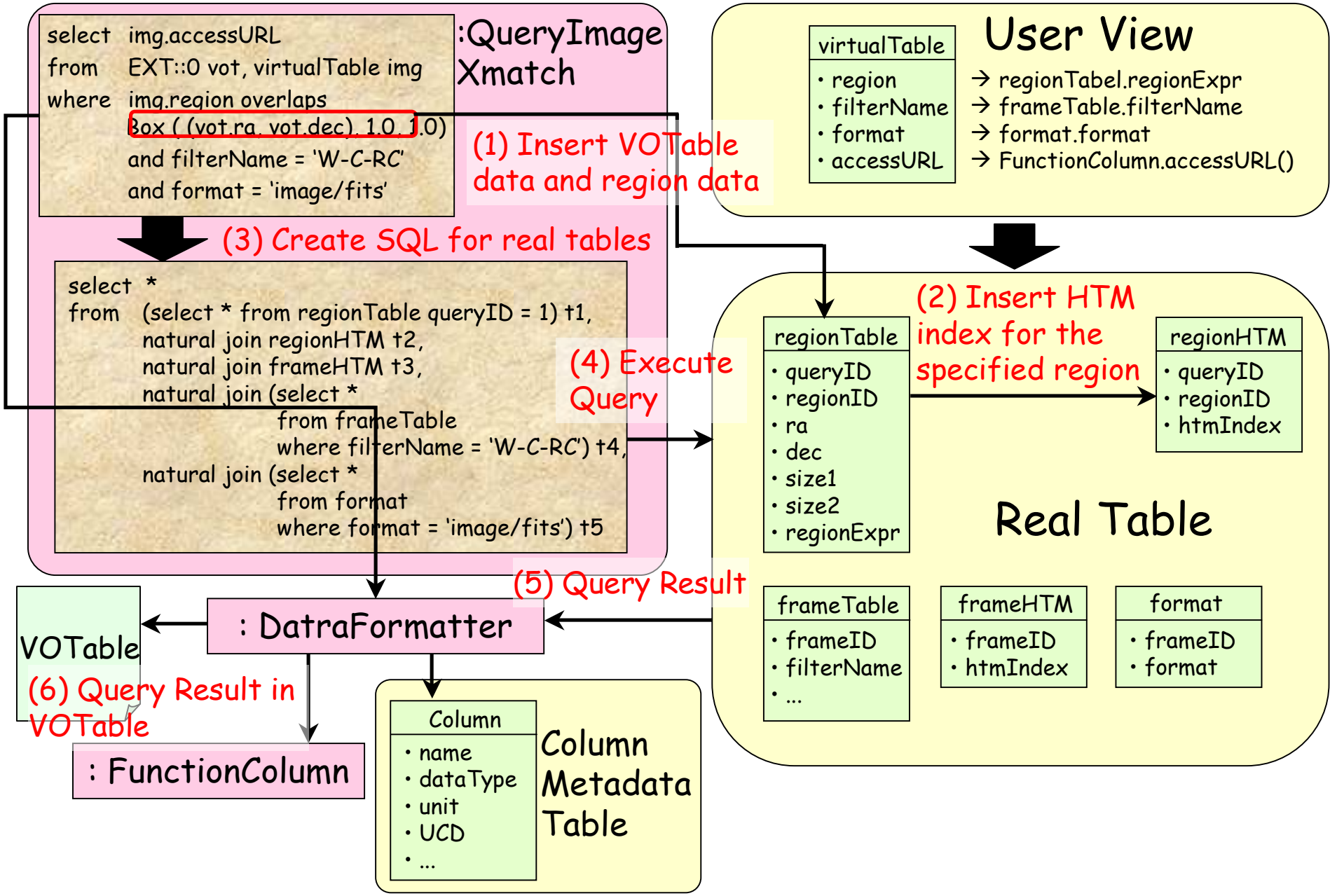
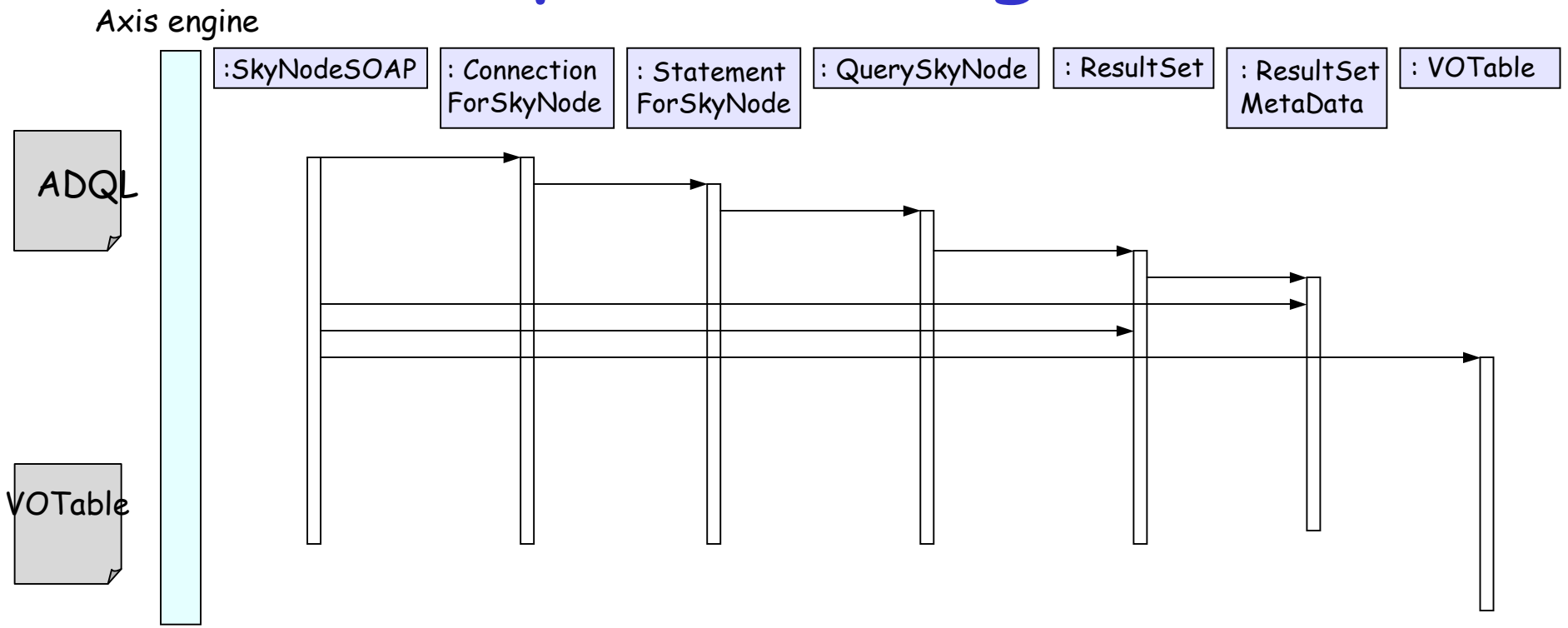


Image Data Xmatch Query with VOTable



Sequence Diagram



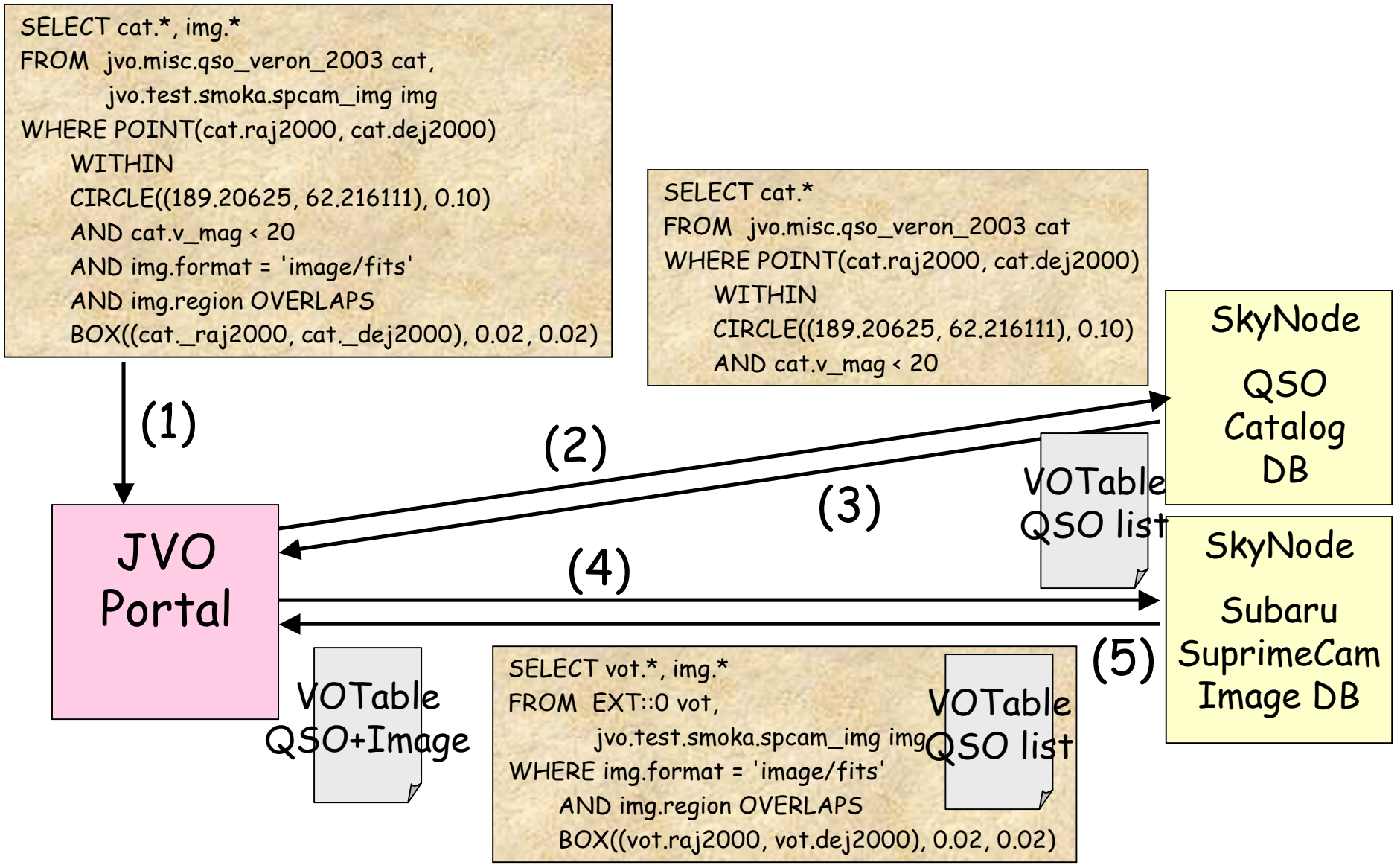
Science Use Case 2
QSO/Galaxy Clustering Study

What we can learn from the QSO/Galaxy clustering ?

- Origin of the large scale structure
  - QSO is a tracer of high density regions in the universe ← hierarchical clustering model
  - Comparison between the observation and theoretical prediction is required.
- Origin of the QSO activity
  - Why is the QSO so powerful.
  - Test of the galaxy merger model

Work flow for studying the QSO/Galaxy clustering

1. Select QSO coordinates from the QSO catalog → Query to the Skynode Catalog Database.
2. Search deep imaging data which covers the QSO regions → Query to the Skynode of Subaru Image Database
3. Create catalog from the imaging data → Invoke the SExtractor Web service.
4. Estimate the distance to the objects around the QSO → Invoke the HyperZ Web service
5. Try Clustering Analysis → Invoke the clustering analysis web service.


SELECT cat.*, img.*
FROM jvo.misc.qso_veron_2003 cat,
jvo.test.smoka.spcam_img img
WHERE POINT(cat.raj2000, cat.dej2000)
WITHIN
CIRCLE((189.20625, 62.216111), 0.10)
AND cat.v_mag < 20
AND img.format = 'image/fits'
AND img.region OVERLAPS
BOX((cat._raj2000, cat._dej2000), 0.02, 0.02)
(1)
JVO Portal
SELECT cat.*
FROM jvo.misc.qso_veron_2003 cat
WHERE POINT(cat.raj2000, cat.dej2000)
WITHIN
CIRCLE((189.20625, 62.216111), 0.10)
AND cat.v_mag < 20
(2)
(3)
SkyNode
QSO Catalog DB
VOTable QSO list
(4)
(5)
SkyNode
Subaru SuprimeCam Image DB
VOTable QSO+Image
SELECT vot.*, img.*
FROM EXT::0 vot,
jvo.test.smoka.spcam_img img
WHERE img.format = 'image/fits'
AND img.region OVERLAPS
BOX((vot.raj2000, vot.dej2000), 0.02, 0.02)
VOTable QSO list


QSO-Galaxies Search - Mozilla Firefox
ファイル(F) 編集(E) 表示(V) 移動(G) ブックマーク(B) ツール(T) ヘルプ(H)
QSO-Galaxies Search
Status | Registry | Search | Result | Database | QSO Search | Image Viewer | Logout
⇒ Query | Catalog | PhotoZ
Data Search
Search
ID for your query
Observation Name
Region
RA: 189.2062 [deg]
Dec: 62.21611 [deg]
Radius: 0.2 [deg]
Image Size: 1.0 [arcmin]
Brightness
V_mag between 10.0 [mag] and 20.0 [mag]
Redshift
z between 0.0 [mag] and 2.0 [mag]
QSO Catalog Search
Service For QSO
ivo://ivo/misc
Table For QSO
qso veron 2003
Image Data Service Search
Service For Image
ivo://ivo/test/smoka
Table For Image
spcam img
Search
User ID
User Name
Group
Last Login
yshirasa
Yuji Shirasaki
jvo
Sat Mar 12 21:36:49 JST 2005

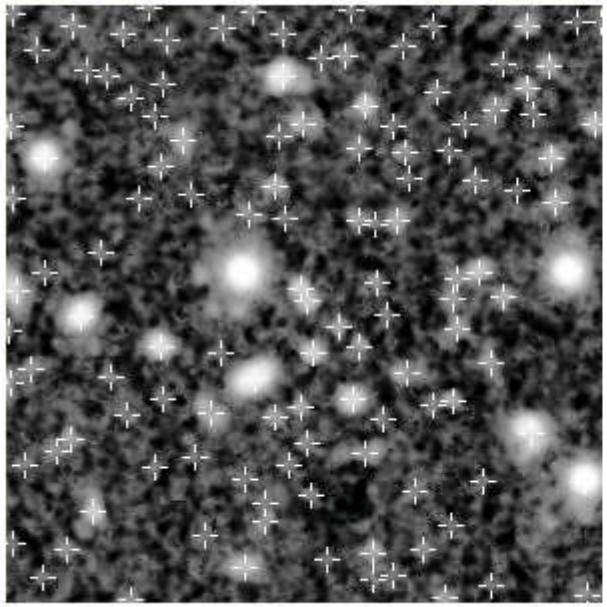
Image Viewer - Mozilla Firefox

ファイル(F) 編集(E) 表示(V) 移動(G) ブックマーク(B) ツール(T) ヘルプ(H)

Image Viewer

Status | Registry | Search | Result | Database | QSO Search | Image Viewer | Logout

Name	Origin	Scale	Contrast
image1	http://hete.mtk.nac	hist	min = 0.0 max = 65535.0 auto = true



Scale : hist eq.

Contrast : auto

min = 0.0

max = 65535.0

Submit

VOTable : catalog

Overplot

User ID	User Name	Group	Last Login
yshirasa	Yuji Shirasaki	jvo	Fri Mar 11 01:06:42 JST 2005

Total memory = 77684kB Used momory = 62095kB (79%)

JVO

JAPANESE VIRTUAL OBSERVATORY

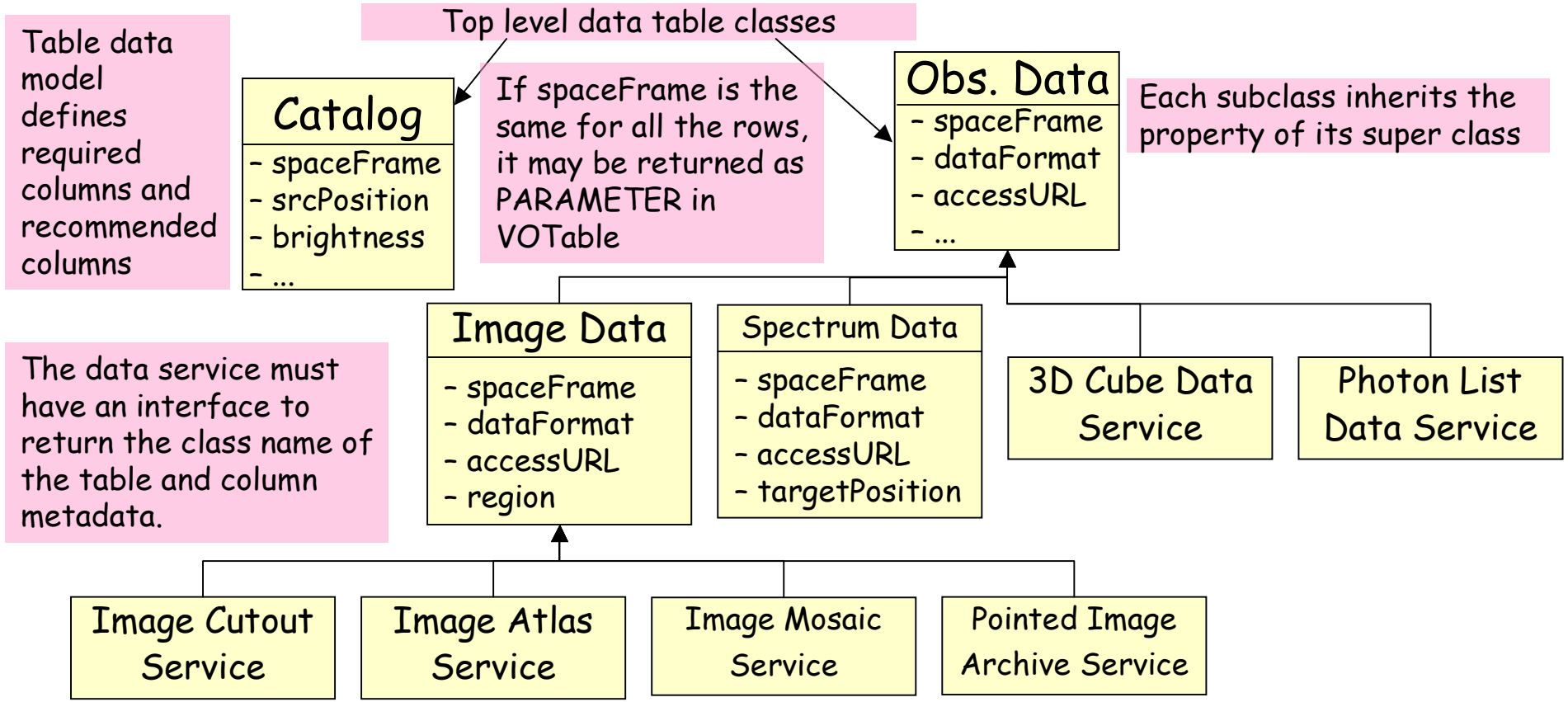
How to treat object data

- Most of the recent RDB support object data type (SQL99).
- relate one object type to one table.
- make it flat

Required Standard

Table Data Model for Astro-Data Service

- Query is described in SQL construct (ADQL).
- Query result is returned in a tabular form (VOTable).
- To write a SQL, table structure (column name) need to be known in advance.
- Table Data Model for each data service are required.



Column metadata

- All the column must have the following attributes:
  - name, datatype, unit, arraysize, width, precision, ucd, utype. These are used in the FIELD element attributes of VOTable. A coordinate column must have an coordinateFrame attribute.
  - column name can be specified by a data provider.
  - datatype must follows the VOTable specification.
  - expression of unit must follows the VOTable spec.
  - if "arraysize" is not defined "*" is assumed for "char" data type and "1" for the other data types.
  - utype is supplied by VO data model naming scheme. For columns not defined in the data model, utype is not required.
  - appropriate ucd is assigned by the data provider.

Data Service Type	Required Columns (utype) (TBR)	Query Spec. of "Where" part
Catalog	Coverage.Location.Sky.Value	<position> within 'STC Region'
Image	Coverage.Region.Sky Data.Format Data.AccessURL	<region> overlaps 'STC Region' and <format> = 'Image/FITS'
Spectrum	Coverage.Location.Sky.Value Data.Format Data.AccessURL	<position> within 'STC Region' and <format> = 'VOTable'
3D Cube (Space x Spectrum)	Coverage.Region.Sky Data.Projection Data.Format Data.AccessURL	<region> overlaps 'STC Region' and <projection> = 'Image' and format = 'Image/JPEG'
Photon list	Coverage.Region.Sky Data.Projection Data.Format Data.AccessURL	<region> overlaps 'STC Region' and <projection> = 'Spectrum' and format = 'Spectrum/FITS'

Minimum Requirement for Query Spec.

Current ADQL:

→  describe most of the SQL92 syntax and some astro-extension

→  many elements (33) & many data types (69), hard to build a data service. It is helpful to define minimum subset ADQL spec as a base line

- SelectionItemType (13 sub types in ADQL 0.8)
  - Only AliasSelectionItem and ColumnReferenceType
  - All the column must have alias name
- FromTableType (3 sub types in ADQL 0.8)
  - Only TableType
- SearchType (16 sub types in ADQL 0.8)
  - Only IntersectionSearchType and ComparisonPredType
  - Region Search Criterion can be specified by ComparisonPredType.

Minimum subset of ADQL for basic Skynode Service

Element: 33 (full) → 12 (basic)

Simple Type: 13 (f) → 4 (b)

Complex Type: 56 (f) → 12 (b)

Fundamentral Type

xs:unsignedInt
xs:string(*)
xs:double(*)
xs:long(*)

Simple Type

aggregateFunctionNameType
allOrDistinctType
binaryOperatorType
comparisonType(*)
jointTableQualifierType
mathFunctionNameType
orderDirectionType
trigonometricFunctionNameType
unaryOperatorType

Element

Allow
Arg(*)
Column
Condition(*)
EndComment
Expression(*)
From(*)
GroupBy
Having
InTo
Item(*)
Literal(*)
Name
Nature
Order
OrderBy
Params
Pattern
Qualifier
Region(*)
Restrict
Select(*)
SelectionList(*)
Set
Sigma
StartComment
Table(*)
TableName
Tables
Unit(*)
Where(*)
fromTableType
selection

Complex Type

ArrayOfFromTableType
ConstantListSet
aggregateFunctionType
aliasSelectionItemType(*)
allSelectionItemType
archiveTableType
atomType(*)
betweenPredType
binaryExprType
closedExprType
closedSearchType
columnReferenceType(*)
comparisonPredType(*)
dropTableType
exclusiveSearchType
fromTableType
fromType(*)
functionType
groupByType
havingType
includeTableType
inclusionSetType
inclusiveSearchType
integerType(*)
intersectionSearchType(*)
intoType
inverseSearchType
joinTableType
likePredType
literalType(*)
mathFunctionType
notBetweenPredType
notLikePredType
numberType
orderExpressionType
orderOptionType
orderType
realType(*)
regionSearchType
scalarExpressionType
searchType
selectType
selectionItemType
selectionLimitType
selectionListType
selectionOptionType
stringType(*)
subQuerySet
tableType(*)
trigonometricFunctionType
unaryExprType
unionSearchType
userDefinedFunctionType
whereType(*)
xMatchTableAliasType
xMatchTyp

stc-v1.20.xsd (element name)

AZ_EL
AstroCoordArea
AstroCoordSystem
BARYCENTER
BODY
CARTESIAN
CatalogEntryLocation
Center
Coord2VecInterval
Coord3VecInterval
CoordArea
CoordFlavor
CoordFrame
CoordInterval
CoordOrigin
CoordScalarInterval
CoordSys
CustomRefPos
CustomSpaceRefFrame
DopplerDefinition
ECLIPTIC
EMBARYCENTER
Equinox
FK4
FK5
File
Frame
GALACTIC_CENTER
GALACTIC_I
GALACTIC_II
GEOCENTER
GEO_C
GEO_D
GSE
GSM
GenericCoordFrame
HCD
HCI
HEE
HEEQ
HELIOCENTER
HGC
HiLimit
HiLimit2Vec
HiLimit3Vec
ICRS
JUPITER
JUPITER_C_III
JUPITER_G_III
LOCAL_GROUP_CENTER
LSR
LSRD
LSRK
LUNA_C
LUNA_G
LoLimit
LoLimit2Vec
LoLimit3Vec
MAG
MARS
MARS_C
MARS_G
MERCURY
MERCURY_C
MERCURY_G
MOON
NEPTUNE
NEPTUNE_C_III
NEPTUNE_G_III
Name
ObsDataLocation
ObservationLocation
ObservatoryLocation
OffsetCenter
PLUTO
PLUTO_C
PLUTO_G
POLAR
PixelCoordArea
PixelCoordFrame
PixelCoordSystem
PixelSpace
PlanetaryEphem
Pole_Zaxis
PositionInterval
RELOCATABLE
Radius
RedshiftFrame
RedshiftInterval
ReferencePosition
Region
RegionFile
SATURN
SATURN_C_III
SATURN_G_III
SM
SPHERICAL
STCResourceProfile
STCmetadata
SUPER_GALACTIC
SearchLocation
SpaceFrame
SpaceRefFrame
SpatialInterval
SpectralFrame
SpectralInterval
Sphere
StartTime
StopTime
TOPOCENTER
TimeFrame
TimeInterval
TimeRefDirection
TimeScale
UNITSPHERE
UNKNOWNFrame
UNKNOWNRefPos
URANUS
URANUS_C_III
URANUS_G_III
VENUS
VENUS_C
VENUS_G
VelInterval
VelocityInterval
VelocitySphere
Xaxis

stc-v1.20.xsd (complexType and simpleType name)

astroCoordAreaType
astroCoordSystemType
catalogEntryLocationType
coord2VecIntervalType
coord3VecIntervalType
coordAreaType
coordFlavorType
coordFrameType
coordIntervalType
coordScalarIntervalType
coordSysType
customRefPosType
customSpaceRefFrameType
fkType
icrsType
obsDataLocationType
observationLocationType
observatoryLocationType
pixelCoordAreaType
pixelCoordSystemType
pixelSpaceType
positionIntervalType
redshiftFrameType
redshiftIntervalType
referencePositionType
regionFileType
regionType
searchLocationType
spaceFrameType
spaceRefFrameType
spatialIntervalType
spectralFrameType
spectralIntervalType
sphereType
stcDescriptionType
stcMetadataType
stcResourceProfileType
stdRefPosType
timeFrameType
timeIntervalType
velocityIntervalType
velocitySphereType
coordEquinoxType
dopplerDefinitionType
planetaryEphemType

region-v1.20.xsd

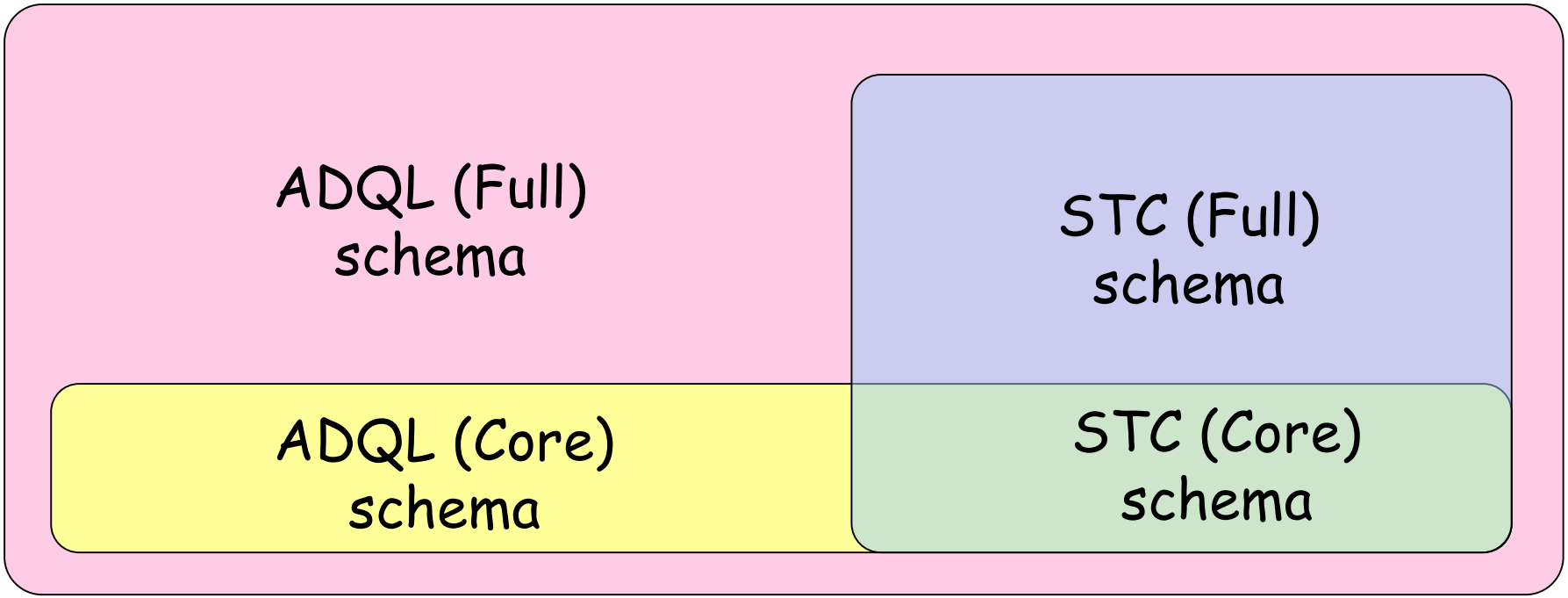
- AllSky
- Box
- Center*
- Circle*
- Constraint
- Convex
- ConvexHull
- Ellipse
- Intersection
- MinorRadius
- Negation
- Offset
- Point
- Pole
- Polygon
- PosAngle
- PosAngle1
- PosAngle2
- Position
- Radius*
- Region
- Sector
- Size
- SkyIndex
- SmallCircle
- Union
- Vector
- Vertex
- boxType
- circleType*
- constraintType
- convexHullType
- convexType
- ellipseType
- intersectionType
- negationType
- polygonType
- regionType*(abstract)
- sectorType
- shapeType*(abstract)
- skyIndexType
- smallCircleType
- unionType
- vertexType

coords-v1.20.xsd

- AbsoluteTime
- AstroCoords
- CError
- CError2
- CError3
- CPixSize
- CPixSize2
- CPixSize3
- CResolution
- CResolution2
- CResolution3
- CSize
- CSize2
- CSize3
- CValue
- CValue2
- CValue3
- CoordFile
- CoordValue
- Coordinate
- Coords
- Error
- Error2
- Error2Matrix
- Error2PA
- Error2Ref
- Error3
- Error3Matrix
- Error3PA
- Error3Ref
- ErrorRef
- FITSFile
- FITSPosition
- FITSRedshift
- FITSSpectral
- FITSTime
- FITSVelocity
- ISOTime
- ISOTimeRef
- JDTime
- JDTimeRef
- MJDTime
- MJDTimeRef
- Name
- PixSize
- PixSize2
- PixSize2Matrix
- PixSize2PA
- PixSize2Ref
- PixSize3
- PixSize3Matrix
- PixSize3PA
- PixSize3Ref
- PixSizeRef
- PixelCoordinate
- PixelCoords
- PosAngle
- PosAngle1
- PosAngle2
- Position
- Position1D
- Position2D
- Position3D
- Redshift
- RelativeTime
- Resolution
- Resolution2
- Resolution2Matrix
- Resolution2PA
- Resolution2Ref
- Resolution3
- Resolution3Matrix
- Resolution3PA
- Resolution3Ref
- ResolutionRef
- ScalarCoordinate
- Size
- Size2
- Size2Matrix
- Size2PA
- Size2Ref
- Size3
- Size3Matrix
- Size3PA
- Size3Ref
- SizeRef
- Spectral
- StringCoordinate
- Time
- TimeInstant
- TimeOffset
- TimeOffsetRef
- TimeOrigin
- Timescale
- Value
- Value2
- Value2Ref
- Value3
- Value3Ref
- ValueRef
- Velocity
- Velocity1D
- Velocity2D
- Velocity3D
- angleUnitType
- double2Type*
- double3Type
- double4Type
- double9Type
- doubleArrayType*
- posAngleReferenceType
- posUnitType
- spectralUnitType
- timeScaleType
- timeUnitType
- unitType
- velTimeUnitType
- astroCoordsFileType
- astroCoordsType
- astronTimeType
- coordFITSColumnsType
- coordinateType
- coordsType
- fitsType
- pixelCoordsType
- posAngleType
- scalarCoordinateType
- size2Type
- size3Type
- timeCoordinateType
- vector2CoordinateType
- vector3CoordinateType

Proposed ADQL Schema structure

- STC (Full) schema imports STC (Core) schema
- ADQL (Core) imports STC (Core)
- ADQL (Full) imports ADQL (Core) and STC (Full)
- ADQL (Core) and STC (Core) should not be changed. These are critical standard for interoperability.
- Each schema has a difference namespace.



Substitution of Column Name by UCD and UType

- Without knowing the column names, we cannot write an SQL…
- Possible Solution → use UCD or Utype for describing the query condition and selection list.
- Instroduce "ucd" and "utype" attributes to the columnReferenceType

```
<Item xsi:type="columnReferenceType" Name="ra" Table="qso"/>
<Item xsi:type="columnReferenceType" ucd="pos.eq;src" Table="qso"/>
<Item xsi:type="columnReferenceType" utype="Target.pos" Table="qso"/>
```

Example Query

```
<Select xmlns:region="urn:region-core"
        xmlns="http://www.ivoa.net/xml/ADQL/core"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <SelectionList>
    <Item Table="t" Name="*" xsi:type="columnReferenceType"/>
  </SelectionList>
  <From>
    <Table Name="image" Alias="t" xsi:type="tableType"/>
  </From>
  <Where>
    <Condition comparison="overlaps" xsi:type="comparisonPredType">
      <Arg Table="t" Name="region" xsi:type="columnReferenceType"/>
      <Arg xsi:type="circleType" unit="deg" coord_system_id="FK5">
        <region:Center>20.0 30.0</region:Center>
        <region:Radius>1.0</region:Radius>
      </Arg>
    </Condition>
  </Where>
</Select>
```

Permanent namespace

Red characters are user-supplied parameters

Use "comparisonPred" for describing the search condition

Only "CircleType" and "AllSky" are defined in region-core schema

```
SELECT t.*
FROM   image t
WHERE  region OVERLAPS
       Circle((20.0,30.0,'FK5'), 1.0)
```

Define Standard Usage of STC Condition

- A basic SkyNode MUST support "comparisonPredType" .
- A basic SkyNode MAY support the other searchType.
- A basic SkyNode MUST recognize an unsupported searchType as a "trueType".
- A basic SkyNode MUST support the following construct:
  - <STC_columnName> <STC_operator> <STC_searchLocationType>
  - STC_operator ::= within, overlaps, outside
    - point within 'Circle ICRS 200 -20 2.0' (Catalog Query)
    - region overlaps 'Circle ICRS 200 -20 2.0' (Image Query)
    - spectrum overlaps 'SpectralInterval A 4000 7000' and observationTime within 'TimeInterval 2004-05-01 2004-05-31' (Specturm Query)

STC-type column must be compared with an object of STC seachLocationType

utype defines which column represents STC.

ADQL-x version of point within 'Circle ICRS 200 -20 2.0'

It might be convenient to define substitution types for frequently used frames such as <SpaceFrame xsi:tye="ICRSFrameType"/>

```
<Condition comparison="within" xsi:type="comparisonPredType">
  <Arg Table="t" Name="point" xsi:type="columnReferenceType"/>
  <Arg xsi:type="searchLocationType">
    <AstroCoordSystem ID="ICRS">
      <SpaceFrame>
        <ICRS/><BARYCENTER/><SPHERICAL coord_naxes="2"/>
      </SpaceFrame>
    </AstroCoordSystem>
    <AstroCoordArea ID="SearchRegion" coord_system_id="ICRS">
      <Region>
        <Circle unit="deg"> <Center>200 -20</Center> <Radius>2.0</Radius> </Circle>
      </Region>
    </AstroCoordArea>
  </Arg>
</Condition>
```

Which frame should be defined as a "must be supported" frame

Returned VOTable

- FIELD id → If alias name is specified "adql:aliasName", otherwise "adql:tableAliasName.columnName".
- FIELD name → free
- STC metadata for STC column
  - If each record in the STC column has a different STC metadata, STC metadata column must be included. Othewise STC metadata must be included as PARAMETER.
- Some of the columns must always be included in the output VOTable. These columns are defined in SIAP, SSAP and SXAP document.

Problem encountered in implementation

Problems encountered in implementation (1)

- Name space problem (as of 2005 Jan)
  - JVO → ADQL v0.8 + VOTable v1.1 + STC v1.1
  - NVO → ADQL v0.74 + VOTable <v1.0 + NVO-STC
  - External interface → ADQL v0.74, VOTable v1.0
  - Internal interface → ADQL v0.8, VOTable v1.1
  - Namespace exchanger
- Possible Solutions:
  - Define a core part of ADQL as a minimum subset and assign a permanent namespace. Never update, never change the namespace of the core part.

Problems encountered in implementation (2)

- XML→ Java deserialization in AXIS
  - With the standard usage of AXIS, an XML document (ADQL, VOTable) is, as a default, deserialized to Java objects.
  - Server memory is easily exhausted.
  - Even several hundreds records of VOTable suffers out of memory error.
- Possible Solution :
  - Stop auto-deserialization of AXIS (learned from an AstroGrid person)
  - return VOTable as an attachment
  - return a reference URL to retrieve the VOTable

Problems encountered in implementation (3)

- Usage of VOTable
  - id and name attribute → what should be filled ?
  - Where column alias name should be put. This information is required for post-search processing on portal side.
  - Information on the origin of the column data should be kept anywhere in VOTable. Where ?

Problems encountered in implementation (4)

- Complexity of ADQL & STC object
- Number of defined element
  - ADQL v0.8 → 33
  - STC v1.20 → 250
- Number of defined ComplexType and SimpleType
  - ADQL v0.8 → 69
  - STC v1.20 → 88

Problem for interoperability

- Column name must be known in advance for writing ADQL.
- We can get column names by "Columns" interface and write ADQL, but it requires human intervention.
- A possible solution:
  - use UCD or Utype for specifying a column
  - Introduce "ucd" and "utype" attributes to the columnReferenceType

```
<Item xsi:type="columnReferenceType" Name="ra" Table="qso"/>
<Item xsi:type="columnReferenceType" ucd="pos.eq;src" Table="qso"/>
<Item xsi:type="columnReferenceType" utype="Target.pos" Table="qso"/>
```

  - If the specified utype or ucd is not found in the queried table,
    - → ignore the condition for that column
    - → return PARAMETER of "NaN" for that column