JVO Skynode Implementation Experience

Yuji SHIRASAKI

National Astronomical Observatory of Japan, JVO

VOQL session

Contents

- Introduction of JVO SkyNode toolkit
  - used free software
  - architecture of JVO skynode
- Problems in implementation and interoperability
  - XML → Java deserialization problem in AXIS
  - Namespace problem ADQL, VOTable, STC
  - Usage of VOTable → id, name attributes ...
  - Complexity of ADQL and 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 (not presented, as a future work)
  - ...

Development of the JVO SkyNode Toolkit

- Primary aim:
  - to provide a reference implementation for every kind of data service using ADQL & VOTable interface
- Supported DBMS:
  - aimed to be independent on the type of DBMS, but still PostgreSQL native SQL is used...
  - The only requirement is availability of JDBC driver.
- Restrictions:
  - Not all the ADQL syntax are supported.
  - String representation of ADQL is JVOQL.
- Experimental Release:
  - http://jvo.nao.ac.jp/download/skynode-toolkit/

What can be done with the toolkit ?

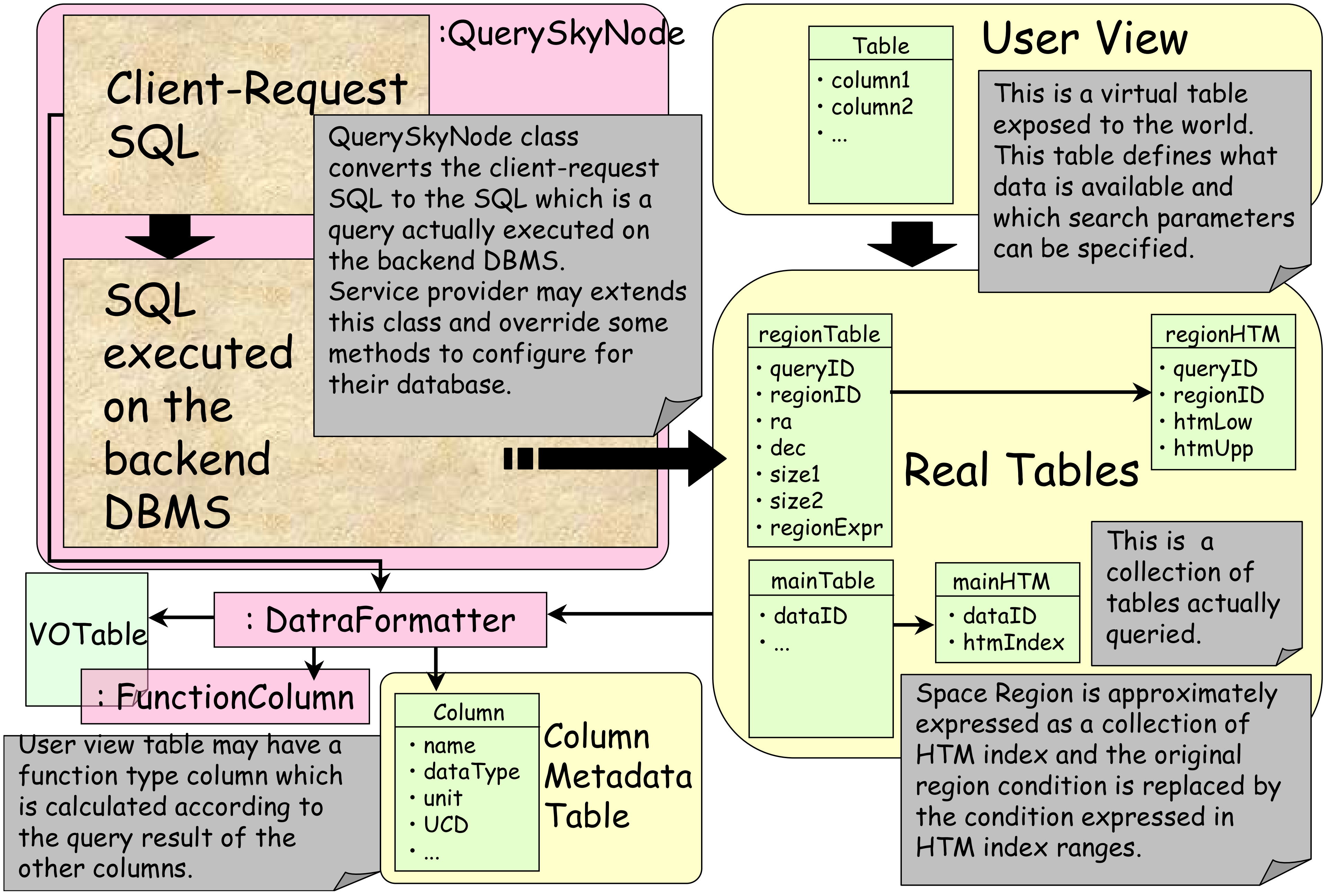
- Catalog data query
- Catalog data cross match query using VOTable
- Image data query
- Image data cross match query using VOTable
- Spectrum data is not supported, but the frame work will be the same as that of Catalog and Image. → next work
- You can build a sample SkyNode service.

Software used

- Tomcat 4.1.31 ---- servlet container
- Axis 1.2RC1 ---- web service engine
- J2SDK 1.4.2 ---- Java compiler & library
- Ant 1.6.1 ---- Java-based build tool
- JavaCC 3.2 ---- parser generator for Java
- JAXB v1.0.3-b18-fcs ---- XML↔Java conversion
- PostgreSQL 7.4.7 ---- DBMS
- Java HTM library (JHU) ---- spherical indexing
- Java FITS library (HEASARC) ---- FITS IO lib
- ...

Architecture

JVO SkyNode Toolkit Flow Chart



QuerySkyNode class converts the client-request SQL to the SQL which is a query actually executed on the backend DBMS. Service provider may extends this class and override some methods to configure for their database.

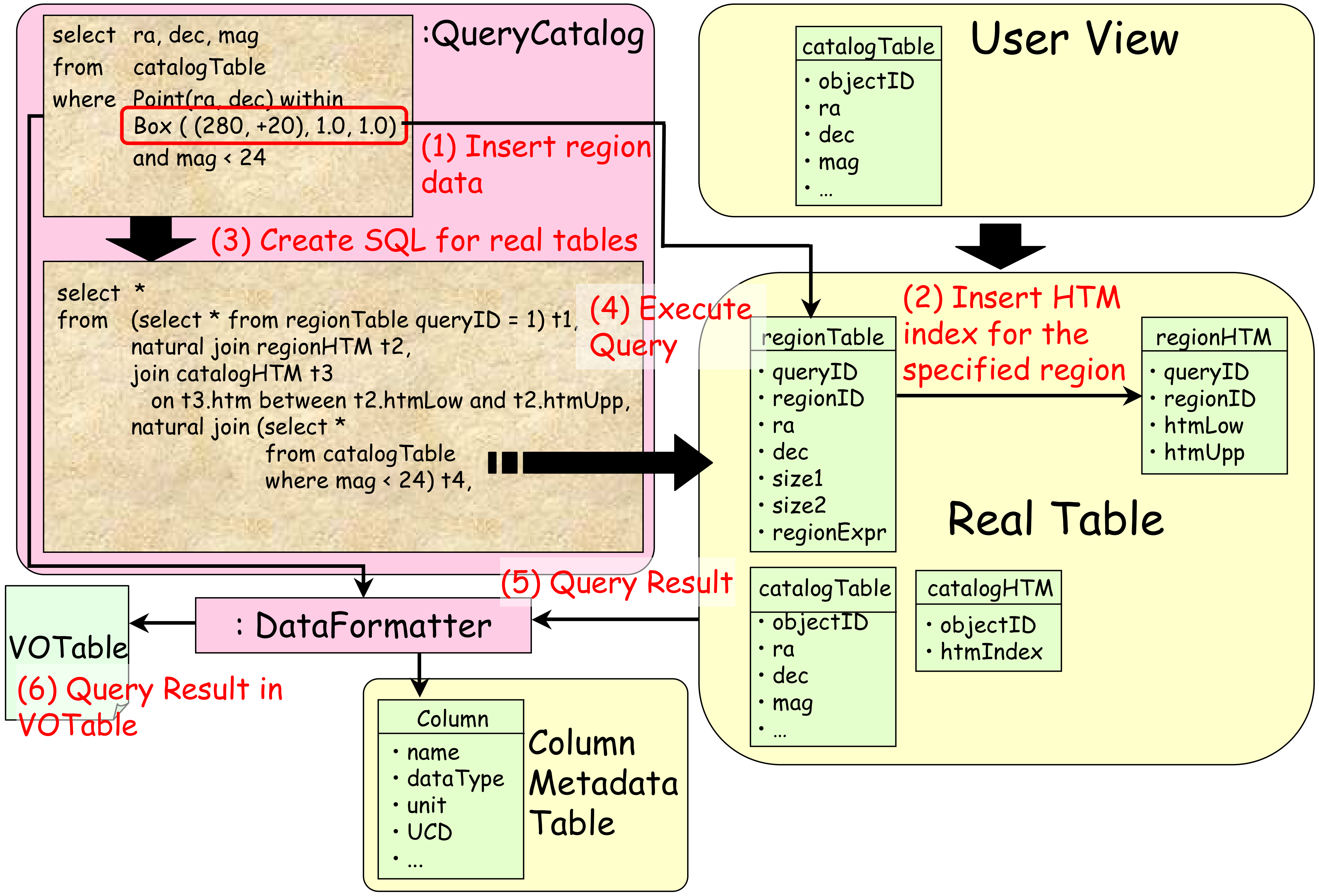
This is a virtual table exposed to the world. This table defines what data is available and which search parameters can be specified.

This is a collection of tables actually queried.

Space Region is approximately expressed as a collection of HTM index and the original region condition is replaced by the condition expressed in HTM index ranges.

User view table may have a function type column which is calculated according to the query result of the other columns.

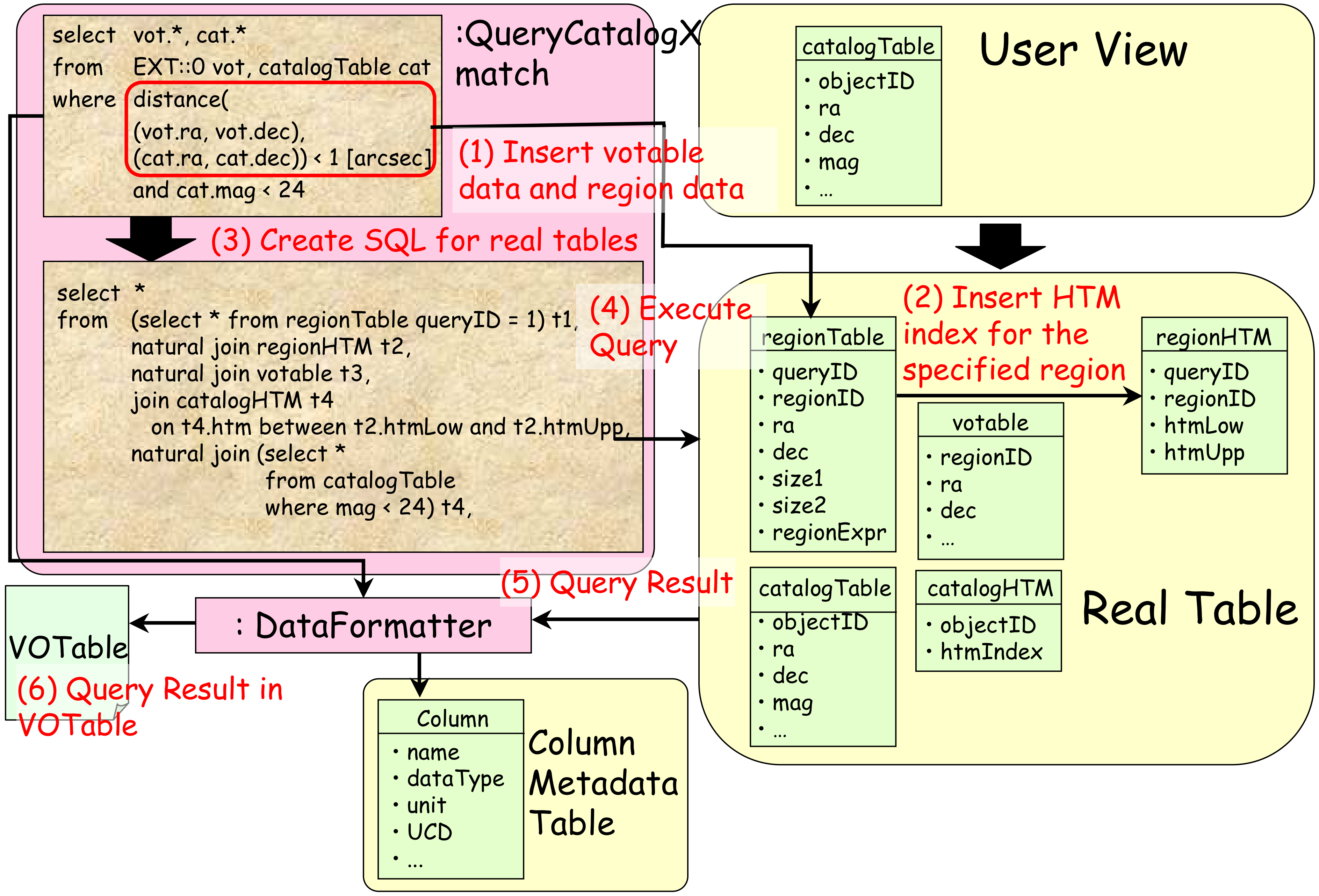
Catalog Data Query by ADQL



```
select  ra, dec, mag
from    catalogTable
where   Point(ra, dec) within
        Box ( (280, +20), 1.0, 1.0)
        and mag < 24
```

```
select  *
from    (select * from regionTable queryID = 1) t1,
        natural join regionHTM t2,
        join catalogHTM t3
          on t3.htm between t2.htmLow and t2.htmUpp,
        natural join (select *
                      from catalogTable
                      where mag < 24) t4,
```

Catalog Data Xmatch Query with VOTable



```
select  vot.*, cat.*
from    EXT::0 vot, catalogTable cat
where   distance(
        (vot.ra, vot.dec),
        (cat.ra, cat.dec)) < 1 [arcsec]
        and cat.mag < 24
```

```
select  *
from    (select * from regionTable queryID = 1) t1,
        natural join regionHTM t2,
        natural join votable t3,
        join catalogHTM t4
          on t4.htm between t2.htmLow and t2.htmUpp,
        natural join (select *
                          from catalogTable
                          where mag < 24) t4,
```

Problem encountered in implementation and problem for interoperability

Problems encountered in implementation (1)

- XML→ Java deserialization in AXIS
  - With the standard usage of AXIS, an XML document (e.g. 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 :
  - Don't use auto-deserialization of AXIS (suggestion from an AstroGrid person), treat the SOAP message as DOM.
  - return VOTable as an attachment
  - return a reference URL to retrieve the VOTable

Problem in implementation (2)

- Usage of VOTable is not clear
  - id and name attribute → what should be filled ?
  - Where column alias name should be put. This information might be used for post-search processing on portal side.
  - Location where a table name and an alias table name are put.
  - Information on the origin of the column data should be kept anywhere in VOTable.

Problems for interoperability (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
- Temporal workaround
  - External interface → ADQL v0.74, VOTable v1.0
  - Internal interface → ADQL v0.8, VOTable v1.1
  - Namespace exchanger
- Complexity of ADQL and STC object
  - ADQL → 33 elements, 69 types
  - STC → 250 elements, 88 types
- 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.

Minimum subset of ADQL

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

Complex Type

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

Problem for interoperability (2)

- 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

Summary

- Experimental release of JVO SkyNode toolkit
  - http://jvo.nao.ac.jp/download/skynode-toolkit/
  - Support for Catalog query, Image query
- Some Proposals
  - Need minimum subset of ADQL and STC
  - Minor update on ADQL: ucd and utype attributes to the ColumnReferenceType.
  - Usage of VOTable. Location where column name, column alias name, table name and table alias name are described.
  - Error message (for future work)