VOQL WG

Yuji Shiasaki

National Astronomical Observatory of Japan

IVOA Plenary

VOQL

Agenda

- Session 1 (Tue 9:00-10:30) ADQL spec.
  - Current Status of ADQL WD (Maria)
  - Survey of SQL compliance of DBMSs (Masahiro)
  - Discussion Topics (Yuji + Any)
    - Syntax of “unit”, “data”, “time”, “timestamp”, “timezone”, “region”
    - Default coordinate frame and unit of region size
    - Xmatch function syntax
    - Table alias & qualified column
    - Replace DBMS dialect with the SQL standard
  - Update of ADQL schema (Yuji)
    - **ADQL-Core**
    - **ADQL-Full**

Agenda (Cont.)

- Session 2 (Tue 11:00-12:30) SkyNode spec.
  - Current Status of SkyNode WD (Maria)
  - Discussion Topics
    - Cross Match (All)
    - Proposal SkyNode classification (Yuji)
    - interface update (Yuji)
    - Table data model (Yuji & Pedro)
    - Content of returned votable (Yuji)
    - Metadata (Yuji)
    - How to manage the ADQL versions (Yuji)

Agenda (Cont.)

- Session 3 (Fri 11:00-12:30) Implementatoin
  - NVO (Maria) : Issues with OpenSkyQuery
  - JVO (Yuji) : SIAP & SkyNode integration
  - ESAC (Aurelien) : Data Model aware SkyNode
  - CDS (someone on behalf of Andre) : VizeR SkyNode
  - AstroGrid (?) : Async data service

Meeting Goal (Primary)

- Agreement on ADQL-Core and Full schema
  - The schema will be tested until the next IVOA meeting, if it is agreed.
- Finalize the cross match debates
  - What level of cross match functionality is required ?
  - Angular distance based selection
  - Chi2 computation
- Share the developed software and increase the SkyNode services
- Slid of my talk can be found at VOQL wiki page.

VOQL session 1

ADQL

Unit

- Unit is exposed by “column” interface or “columns” table, so basically it is possible to do the unit conversion at the client side.
- Use same syntax as specified in VOTable WD.
  - http://vizier.u-strasbg.fr/doc/catstd-3.2.htx
- Is it required to implement unit conversion for all the units described in the above URL ?
  - CDS has software to do the conversion.
- Support at least the conversion among “deg”, “arcmin”, “arcsec”, “radian”

Date/Time/Timestamp/TimeZone

- Many variety for ISO 8601 (STC) expressions
- standard SQL just covers only a part of ISO8601
- ADQL supports the yellow colored expression ?
- Recommend a client app to support all for user input

- 2006-05-16 (o)
- 2006-05 (x)
- 2006 (x)
- 2006-001 (x)
- 2004-W13-4 (x)

- 10:30:50.012 (o)
- 10:30:50 (o)
- 10:30 (x)
- 10:30.5 (x)
- 10 (x)
- 10.5 (x)

- +09:00 (o)
- Z (x)
- +09 (x)
- +0900 (x)

- 2006-05-16 10:30:50.012 (o)
- 2006-05-16T10:30:50.012 (x)

XMatch Function

- `XMATCH_CHI2('t1 t2 !t3',3.5)`

- `XMATCH_DISTANCE(t1.ra, t1.dec, t2.ra, t2.dec, 1.0 [arcsec]) AND XMATCH_DISTANCE(t1.ra, t1.dec, t3.ra, t3.dec, 1.0 [arcsec])`

Region Syntax

```
Region('<shape> [<frame>] <ra> <dec> <size>')
<shape> ::= BOX | CIRCLE
<frame> ::= FK4, FK5, ICRS, Gala, what else ?
<ra> ::= <numeric literal>
<dec> ::= <numeric literal>
<size> ::= <numeric literal> [ <unit> ]
<unit> ::= deg | arcmin | arcsec
```

- "Sexagecimal" is not allowed
- Recommend a client app to support "Sexagecimal" for user input
- Supported frames and units should be exposed by metadata interface

Default Coordinate Frame and Unit

- When coordinate frame and/or unit are omitted, what default setting should be used ?
  - Service specific frame and unit
  - They should be exposed as metadata.
- It is natural to do a region search on the coordinate frame specific to the table.
  - Simulation data : if we define default frame and unit to e.g. “FK5” and “deg”, it will not be applicable to the simulation data.

Table Alias, Qualified Column

- Table alias name is mandatory
- All the column name must be qualified by table alias name not by table name.
- Recommend that client app allow a user to omit the table alias and column qualifier in a trivial case (single table query), and that the client app gives a default table alias name when submiting ADQL-x.

```
SELECT   ra, dec
FROM     qso
WHERE    Region('Circle 210 30 1.0')
```

Delimited Identifier

- According to the SQL standard, double quotations are used to specify the delimited identifier.
- Current ADQL uses “[“ and “]” (dialect of SQLServer)
- Why not use the SQL standard
- This was discussed at the previous IVOA meeting, and there was no claim to use the standard SQL.

Select Into

- "Select into" is a dialect of SQLServer
- "Create table as (<select statement>)" is defined as a SQL standard (SQL99 ?)
- Why not use the SQL standard ?

ADQL schema is split into two schemas

- ADQL-Core and ADQL-Full
- ADQL-Core schema conforms to the ADQL core specification
- ADQL-Core schema is aimed to be used for interoperability, update cycle will be longer than ADQL-Full (>10 years ?).
- ADQL-Full schema is aimed to be used for implementing advanced query functionality. Update cycle should be as long as possible (>5 year).

ADQL schema update

- Verbose ComplexTypes are replaced by one ComplexType → simplified
- Added SQL syntax (natural join, join using, subquery, exists, any, all) → higher functionality
- 56 complexType, 9 simpleType (ADQL 1.0)
- 53 complexType, 3 simpleType (ADQL 1.041)
- 35 complexType, 0 simpleType (ADQL Core)
- 1.0 is translatable to 1.041 without loss of information. Core is translatable to 1.041 w/o loss of information

ADQL schema update

- removed type definitions

  binaryOperatorType, unaryOperatorType, atomType, stringType, trigonometricFunctionType, trigonometricFunctionNameType, mathFunctionType, mathFunctionNameType, aggregateFunctionNameType, comparisonType, archiveTableType, xMatchTableAliasType, includeTableType, dropTableType, xMatchType, notLikePredType, exclusiveSearchType, notBetweenPredType, inverseSearchType, userDefinedFunctionType, ArrayOfFrmoTableType

ADQL schema update (cont.)

- Added complex type:

  xpathReferenceType, nonNumericType,
  subqueryTableType, joinConditionType crossJoin,
  onJoin, naturalJoin, usingJoin,
  booleanValueFunctionType, existsPredType,
  anyPredType, allPredType

ADQL schema update (cont.)

- + selectionLimitType: offset attribute is added
- + fromType: maxOccurs of Table element is changed from "unbounded" to "1"
- + searchType: “not” attribute is added
- + columnReferenceType: CaseSensitive attribute is added, xpathName attribute is removed as xpathReference is introduced.
- + functionType: abstract="true" is removed, Allow element is removed, number of appearance of an Args element changed to “unlimited”, Name attribute is added.
- + aggregateFunctionType: changed to extend scalarExpressionType, Name attribute is added, Allow and Arg elements is added.
- + numberType: unit attribute is added.
- + integerType: type of value attribute is changed from xs:long to xs:integer.
- + tableType: attributes “ShortName”, “Identifier” and “CaseSensitive” are added, “xpathName” is removed
- + joinTableType: “LeftTable” and “RightTable” are added, Qualifier, Tables elements are removed
- + joinTableQualifierType: "_OUTER" suffix is removed, "CROSS" is removed.
- + likePredType: type of Pattern element is changed to nonNumericType.
- + regionSearchType: ???

ADQL schema update (cont.)

- binaryOperatorType:
  - enumeration of strings “+”, “-”, “*”, “/”
  - Removed to allow for service specific operators.
  - The operators that should be supported are described in another document (ADQL WD or note ?)
  - unaryOperatorType (“+”, “-”) and comparisonType (“=“, “<“, “>” …) are also removed for the same reason.
- atomType:
  - just a wrapper of literalType, unit is defined here.
  - Removed for verbosity
  - Unit is defined at NumericType
  - stringType is renamed as nunNumericType to be used for non-numeric type such as timestamp, boolean, spaceCoords, spaceRegion, and a service specific type.

ADQL schema update (cont.)

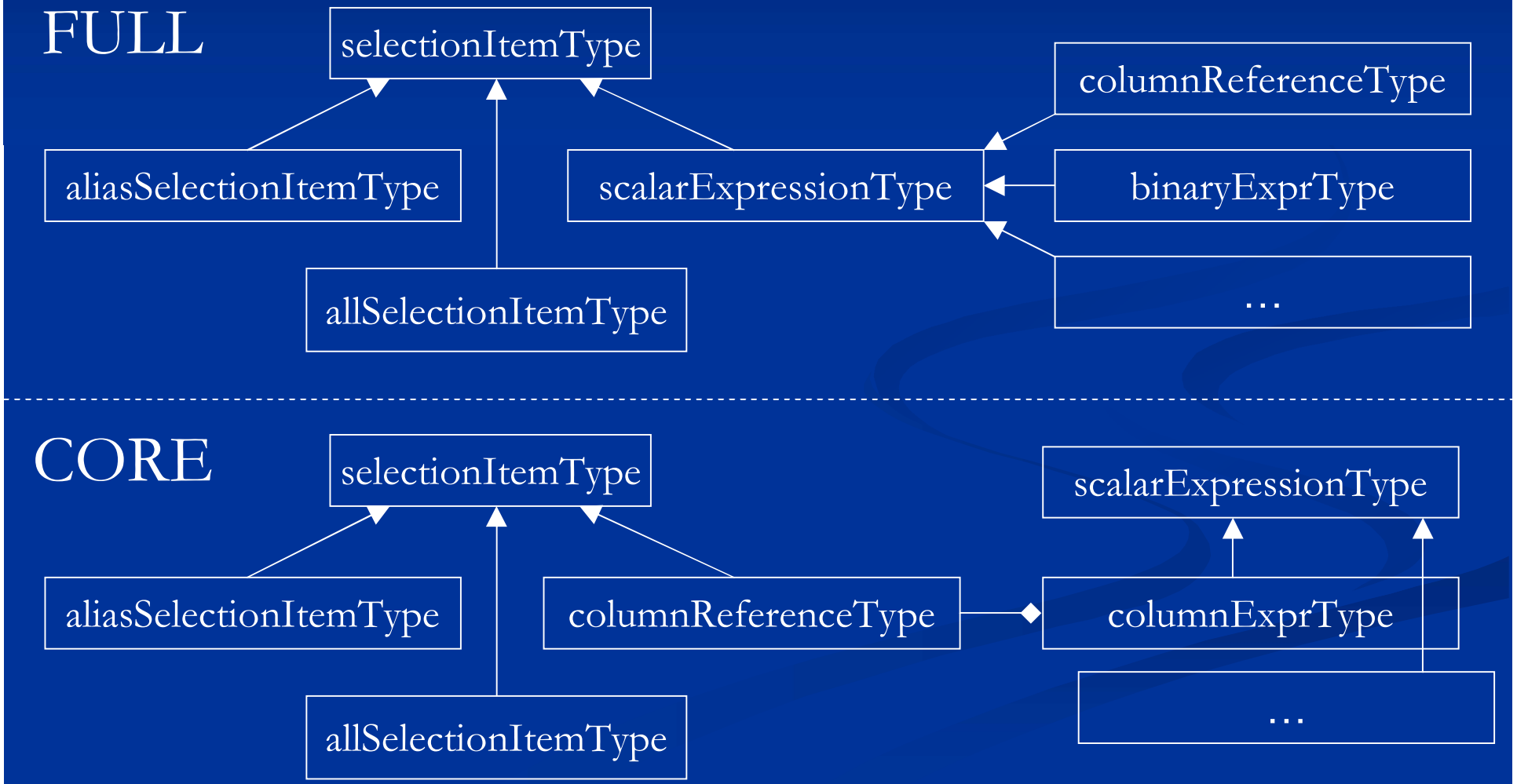
- FunctionType familiy
  - trigonometricFunctionType, mathFunctionType, userDefinedFunctionType are unified to a single FunctionType.
- ArchiveTableType
  - Identifier attribute is added TableType, so this is obsoleted.
- XMatchType family
  - xMatchType, xMatchTableAliasType and so on are removed
  - Xmatch is expressed by a FunctionType wrpped by booleanValueFunctionType

ADQL schema update (cont.)

- NOT family
  - notLikePredType, exclusiveSearchType, notBetweenPredType, inverseSearchType are removed
  - Not attribute is added to the searchType
- EXISTS, ANY, ALL
  - EXISTS (subquery)
  - Column = ANY (subquery)
  - Column = ALL (subquery)

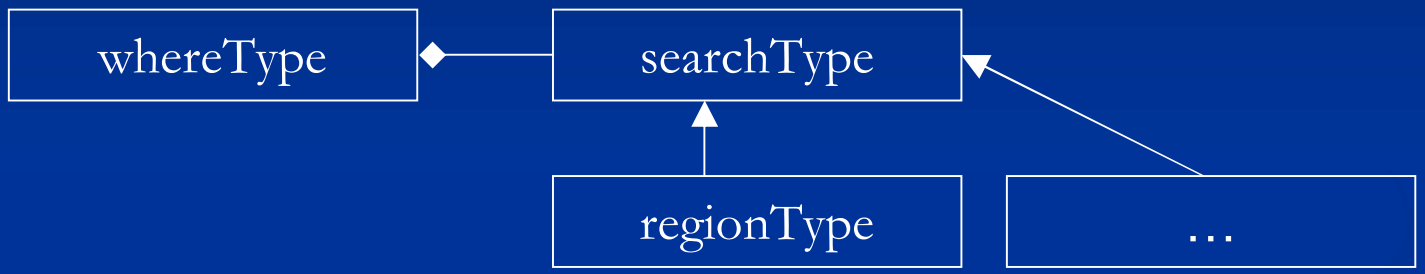
ADQL Core schema

- Structure is slightly different from ADQL full schema.



ADQL Core schema (cont.)

FULL



CORE

Only one region can be specified.

ADQL Core schema (cont.)

- TableJoin family is removed
- Only one TableType is specified.

VOQL session 2

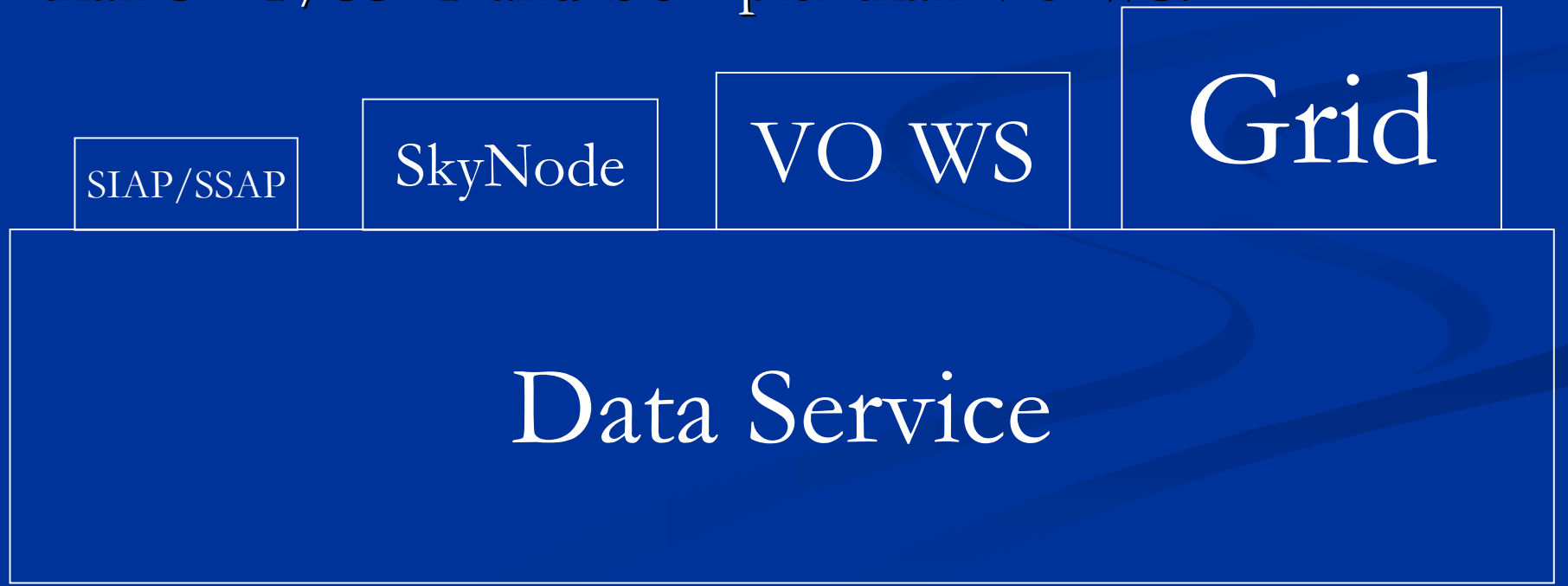
SkyNode

How to live with the VO WS standard

- GWS WG is preparing a common VO IF
  - UWS, CEA, VOStore …
- There will be a more global standard: Grid by GGF
- Will what we are defining in the SkyNode spec will be deprecated ?
  - No. (at least I think so)
  - We are defining SkyNode specific interface, and the interface is simpler and easier to use than the general standard interface.

Hierarchy of Protocol

- Capability of the data service is increased by adapting the higher level protocol
- But complexity is also increased
- Adapt appropriate interface which matches the scale or required visibility of the data service.
- VOQL WG defines the interface that has more capability than SIAP/SSAP and is simpler than VO WS.



Proposal of new interface

- Vodata = performQuery(adqlCore, format)
- Vodata = performQuery(adql, votable, format)
  - There was a xmatch() interface in earlier version but is was hidden by executePlan inerface. It is worthwhile to have this interface independent of executePlan.
- Jobid = performQueryAsync(adql, votable, format, listenerURL)
- Status = performPolling(jobid)
  - "Status" shows whether the query is running or finished. If finished it gives an URL to retrieve the data.
- destroy(jobid)
  - Remove all the resources generated by the job

What should "select into" returns ? empty votable ?

- This query should be used only for performQueryAsync() interface ?

Cross match proposal

- Which algorithm should the xmatch-able skynode support ?
  - Chi2 calculation vs angular distance
- “angular distance” based cross match as a primary algorithm → all the xmatch-able skynode must support this.
- “chi2 calculation” based cross match as an advanced functionality of the xmatch-able skynode.
- Any other algorithms may be supported.
- Supported algorithms (function names) should be exposed by metadata interface

Skynode classification.

- Only the two calssification is not enough : BASIC and FULL
- At least following types will exist:
  - BASIC Skynode
  - FILE UPLOADABLE Skynode
  - Cross match support Skynode
  - ExecutePaln support Skynode
  - Async Skynode

Content of a returned VOTable.

- The order of the FIELD should be the same as the order in the selection list, which enables to access to the data by index id.
- If "*" is specified in the selection list, the order should be decided on the server side.
- All the column metadata should be properly set to the FILED attributes
- "Name" attribute of the FIELD should have a qualified column name. Qualifier should be a table alias name
  - <tableAlias>.<columnName>
- Column metadata that cannot be set to the FIELD attribute may be set by using <VALUES> tag.
  - <VALUES><OPTION name="meta:name" value="value"/></VALUES>
  - This is not the correct usage of <VALUES> tag, but…

Table data model

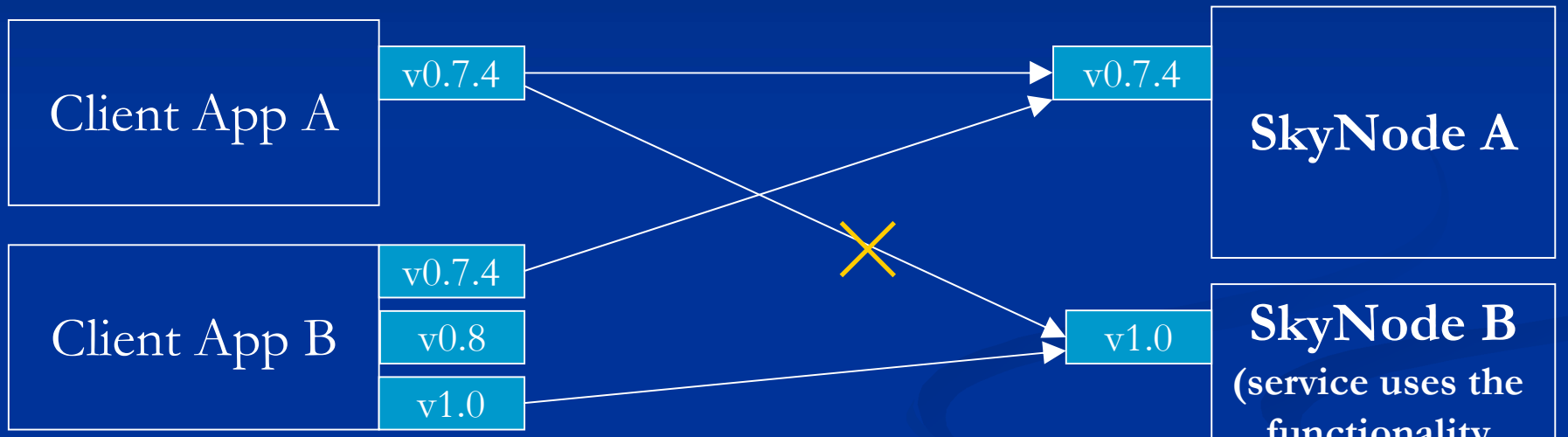
- Define table classes according to the contents of the table
  - General, ObjectCatalog, ObjectBrigthnessCatalog, ObservationCatalog, Image, Spectrum
- For each table class, define columns that must be included. Use utype.
  - General → no requirement
  - ObjectCatalog → utype = id, pos.ra, pos.dec
  - ObjectBrightnessCatalog → id, pos.ra, pos.dec, brightness[i], wavelength_range[i]
  - ObservationCatalog → TBD
  - Image → defined in SIAP
  - Spectrum → defined in SSAP

Metadata: metadata tables vs tables & columns interface.

- Do we need two ways to access to the metadata ?
- Use metadata tables to get more precise information about table and column metadata.
- Use tables and columns interface to get metadata defined as “must provide”.
- Metadata table “tables” and “columns” should have columns that defined as mandatory.
- Metadata table “tables” and “columns” may have columns that is specific to the service.

How to manage the ADQL versions

- Service may be implemented with any public version of ADQL
- Service should expose the supported versions as metadata



- I suggest that the multiple version functionality is implemented on client side, since the number of client app is fewer than the service.

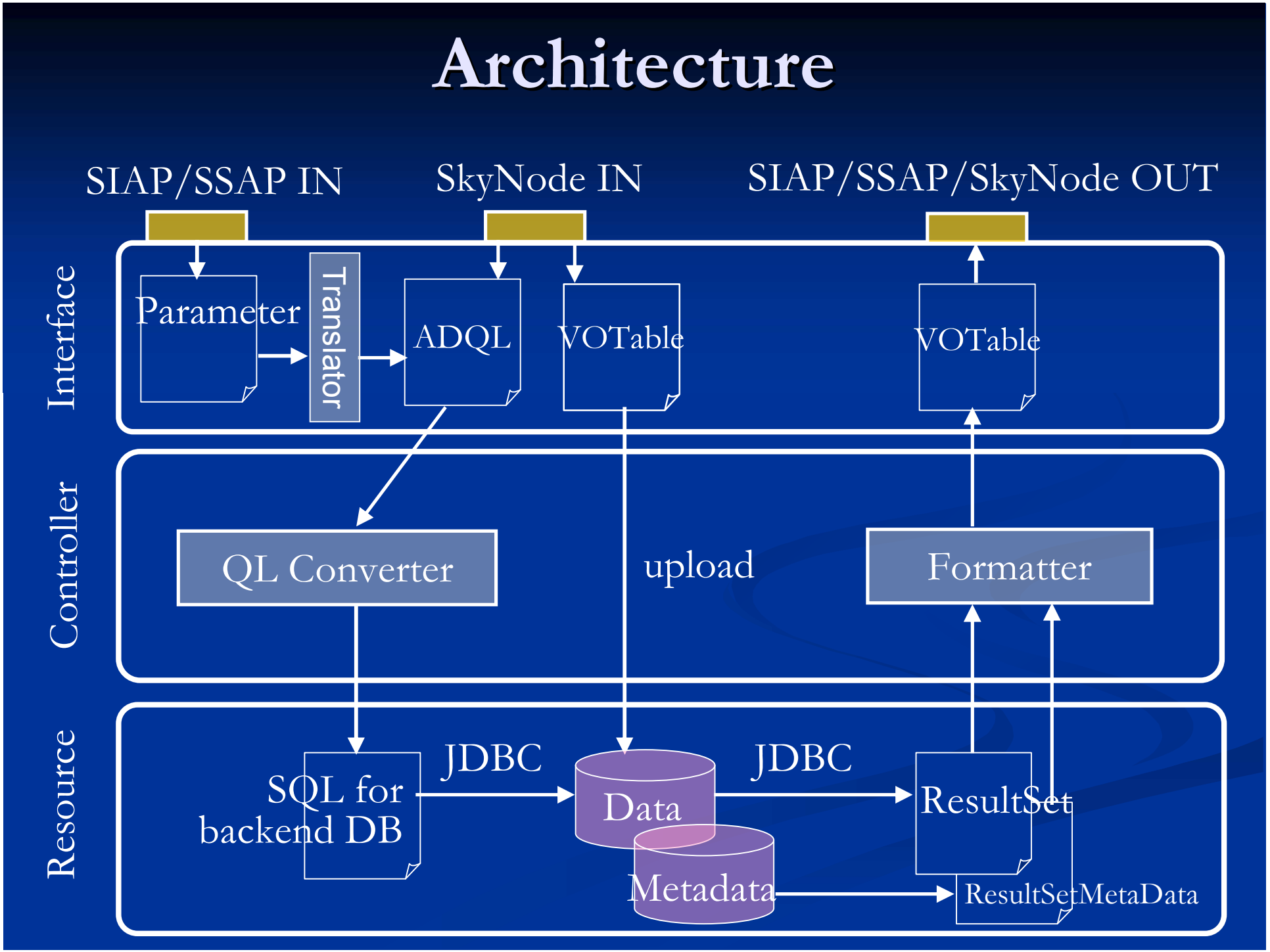
VOQL session 3

Implementation (Under preparation)

SkyNode toolkit

- http://jvo.nao.ac.jp/download/skynode-toolkit/
- What this toolkit can do / provide ?
  - Basic skynode can be easily set up provided that data are already stored on DBMS and JDBC driver is available.
  - Image data service that returns a VOTable conforming to the SIAP.
  - Java SQL parser is included. ADQL-s ←→ ADQL-x.

Architecture



SIAP compliant query

- SIAP query parameters are mapped to table columns.
- Metadata returned as the SIAP query response are mapped to table columns

How to setup

- Read instruction.txt
- Minimum requirement:
  - Java, Tomcat, DBMS (PostgreSQL, MySQL), Skynode TK
- Procedure:
  - Copy jvop3-skynode.war under webapps of tomcat.
  - Create etc, var/log, tmp under the directry /usr/local.
  - Copy jvo.properties, log4jproperties, deploy-template.wsdd at /usr/loca/skynode/etc.
  - Start tomcat.
  - Create metadata database.
  - Deploy the skynode web service.