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 ?
- Define basic unit that must be converted

Data/Time/Timestamp/TimeZone

- There are may variety for ISO 8601 (STC) expressions, and standard SQL expression just covers only a part of ISO8601 expression

- 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)

To put a separator T is not an SQL standard

Sexagecimal

Region syntax

- `Region('<shape> [<frame>] <ra> <dec> <size>')`
- `<shape> ::= BOX | CIRCLE`
- `<frame> ::= FK4, FK5, ICRS, Gala,` what else ?
- `<ra> ::= <numeric literal> | ' <sexagecimal> '`
- `<dec> ::= <numeric literal> | ' <sexagecimal> '`
- `<size> ::= <numeric literal> [ <unit> ]`
- `<unit> ::= deg | arcmin | arcsec |` …
- Supported frames and units should be exposed by metadata interface

Default Coordinate Frame and Unit

- When coordinate frame and/or unit are omitted, how the Skynode should react ?
  - Defaulted to service specified frame and unit, which 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, it will not applicable to the simulation data.
- You should explicitly specify the coordinate frame if you specify the region in your favorite frame

Cross match

- 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

Table & Column Alias

Delimited Identifier

- SQL standard is to use double quotations to specify the delimited identifier.
- Current ADQL uses “[“ and “]”
- Why not use the SQL standard
- This was discussed at the previous IVOA meeting, and there was no claim to use the standard SQL.

Comment

Select Into

- “Select into” is a dialect of SQL server
- “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.

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)
  - 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.)

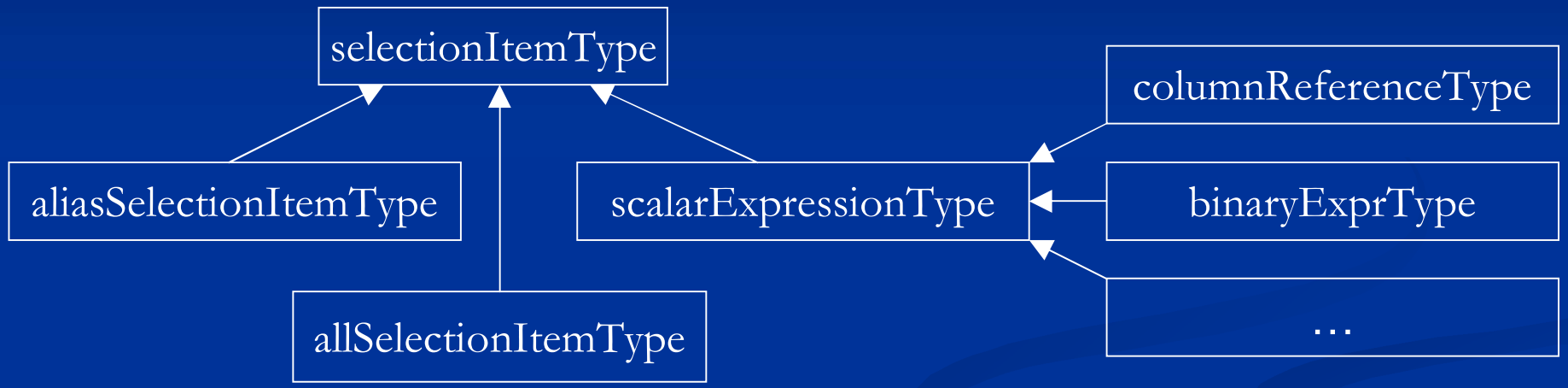
- FunctionType familiy
  - trigonometricFunctionType, mathFunctionType, userDefinedFunctionType are unified to a single FunctionType.
- ArchiveTableType
  - Identifier attribute is added TableType, so this is obsoleted.
- XMatchType family
  - xMatchType, xMatchTableAliasType and so on are removed
 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

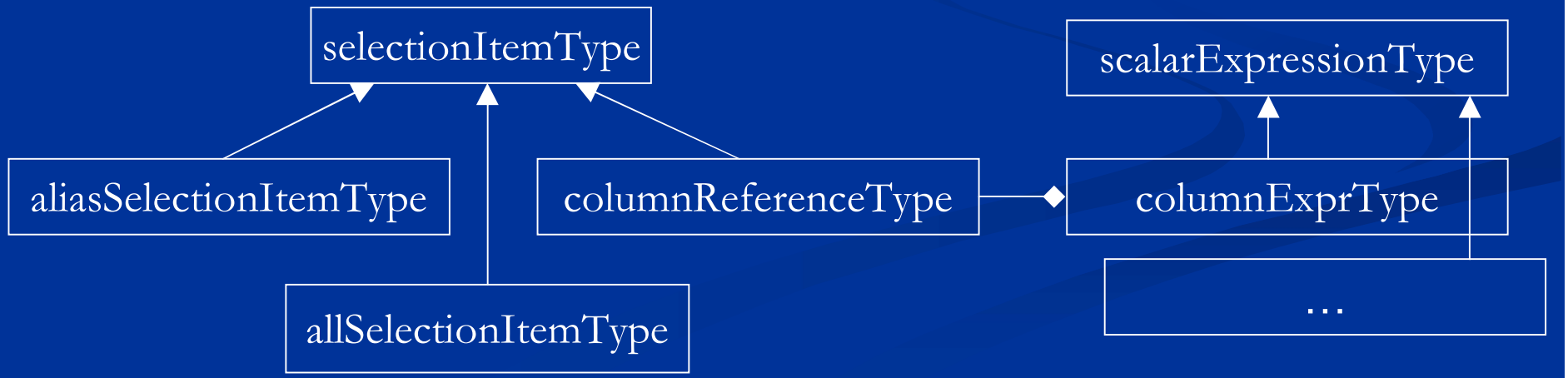
ADQL schema update (cont.)

- NOT family
  - notLikePredType, exclusiveSearchType, notBetweenPredType, inverseSearchType are removed
  - Not attribute is added to the searchType
- EXIST, ANY, ALL

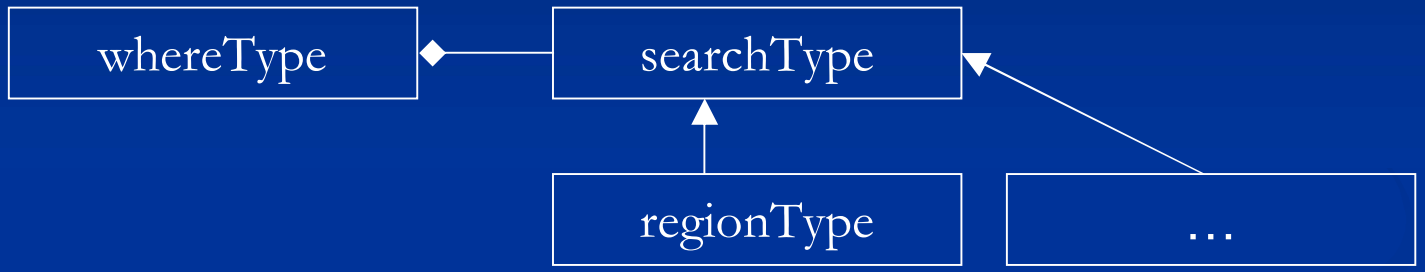
ADQL Core schema

- Structure is slightly different from ADQL full schema.





ADQL Core schema (cont.)



VOQL session 2

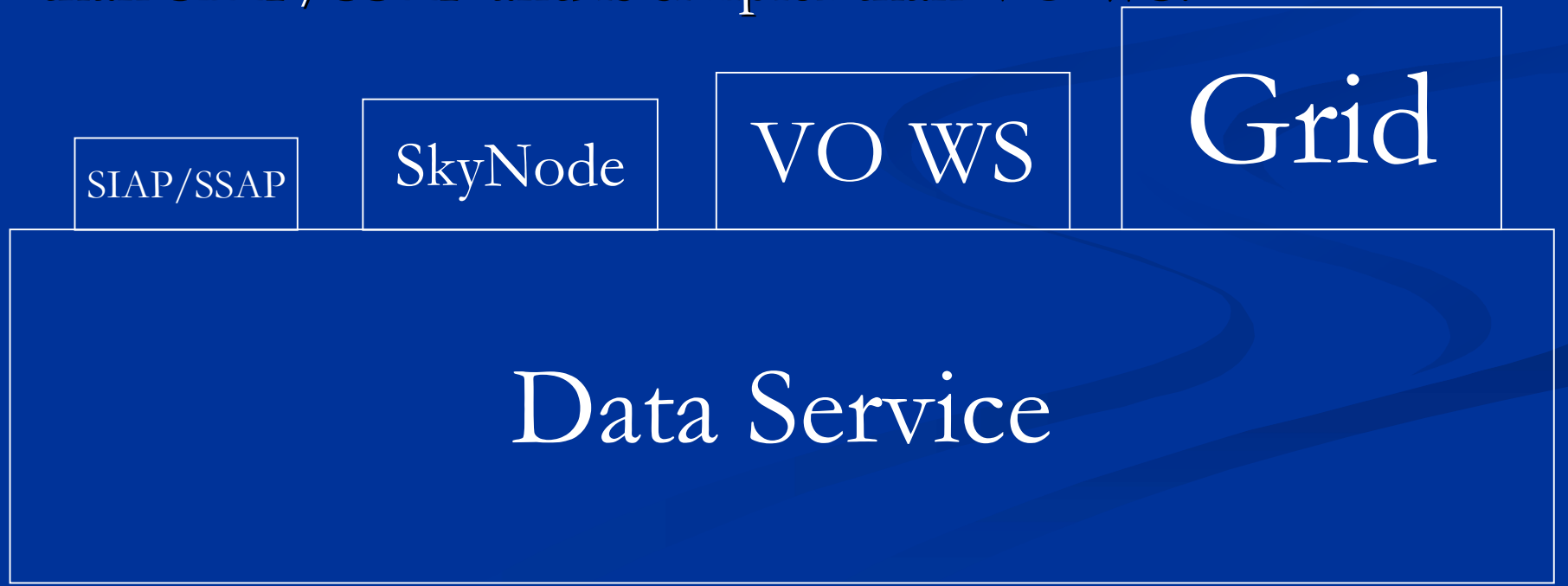
SkyNode

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 more simple and more easy 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 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.

What should "select into" returns ? empty votable ?

- This query should be used only for performQueryAsync() 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 supoort Skynode
  - Async Skynode

content of a returned VOTable.

- The order of the FIELDS 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 propery set to the FILED attributes
- “Name” attribute of the FIELD should have a qualified column name. Qualifier should be a table alias nema
  - <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>

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, wavelength_range
  - ObservationCatalog →
  - Image → defined in SIAP
  - Spectrum → defined in SSAP

Metadata: metadata tables vs tables & columns interface.

- Use metadata tables to get more precise information about table and column metadata.
- Use tables and columns interface to get primary metadata.
- 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.

VOQL session 3

Implementation