IVOA VOQL

Working Group Report

Moscow Interop meeting

Sep 2006

VOQL Group Configuration

Configuration since 27 July 2006:

- Chairman: P. Osuna (ESA-VO)
- Vice-chairman: Y. Shirashaki (JVO)

Thanks to María and Yuji for the work done so far

Current situation and possible evolution

- ADQL and SkyNode specifications intermingled
- VOQL (currently ADQL) should only deal with Language related issues.
- SkyNode – like implementations making use of VOQL should be specified separately
- “Basic” SkyNode functionality is different from “Full” SkyNode functionality → two different concepts arise

Plans for the future

- Separate current ADQL/SkyNode in the following three items:
  - VO Query Language (VOQL)
  - Table Access Protocol (TAP)
  - SkyNode (Full)
- VOQL Group to deal with VOQL specificatioon ***and*** TAP specification (latter, together with DAL group)
- SkyNode specification to be dealt with separately

VOQL Technical Experts Group (VOQL-TEG)

- Based on Registry Tiger Team very positive experience
- Contains one member from each of the institutions mostly involved in VOQL issues
- Reduced group of technical experts to work on:
  - VOQL and TAP Specifications
  - Helper tools whenever needed
  - Reference implementations for the specifications
  - Community support

VOQL-TEG Members

- Alex Szalay (NVO)
- Francois Ochsenbein (CDS)
- Pat Dowler (CVO)
- Kona Andrews (Astrogrid)
- Yuji Shirasaki (JVO)
- Aurelien Stebe (ESA-VO)

Coordination: P. Osuna (**IVOA**)

ADQL comments after WD1.05

- Current concerns (after last working draft)
  - INTO and #UPLOAD: should they both be there? Only INTO? #UPLOAD part of VOSpace-like spec.?
  - Service identifiers in front of table names needed?
  - “Keyword id, delimiter id” only needed because of ADQL/s usage?
  - Xmatch should not be defined inside ADQL
  - Metadata queries are not a good idea
  - No mention of output handling should be done in a language specification doc
  - Why force the usage of table aliases

ADQL comments after WD1.05 (II)

- Why limit one unique table in eventual Core ADQL syntax
- Why limit Core ADQL syntax to “AND” relations in the WHERE?
- OFFSET and TOP: better supported as part of service interface (like the registry does)
- Do we need all types of joins? Shouldn’t natural joins be enough?
- EXIST, ALL, SOME…. What are these things?
- More details on Xpath/utype usage needed
- Allowing for service specific data types is not a good idea
- Aggregate functions should be defined by the service
- “circle” and “box” should be REGION functions

ADQL comments after WD1.05 (III)

- Should ADQL have ADQL/s and ADQL/x? Only ADQL/s? Only ADQL/x?
- ADQL based in SQL:
  - Remove db mntc constructs
- All extensions via “macros” (or “user defined functions”):
  - Allowing more than one DB
  - Region, crossmatch,…
  - Utypes…

ADQL possible scheme

- One Core with simple SQL-like plus maybe REGION
- One single extension allowing for clauses (group by, having, …)
- One single placeholder for User Defined functions. Those are specified in Registered services. For instance:
  - A Crossmatch service (of whichever type) might be Registered as understanding:
    - Core ADQL
    - Extended ADQL
    - UserDefinedFunctions:
      - XMatchChi2
      - PositionalXMatch
    - The definition of the User Defined Funtcions would appear in the specification of the specific service (e.g., in an eventual SkyNode spec)

Table Access Protocol (TAP)

- Name not a big issue, but decided for TAP rather than STAP, as the the protocol should support “S”imple or “C”omplex access
- Will be defined within the VOQL-TEG with support from the DAL (through Doug Tody)
- Interest in having the TAP be able to:
  - Should correctly execute a VOQL query in the core syntax
  - Should make a best attempt to execute queries containing extended constructs which may contain vendor specific items

Table Access Protocol (II)

- Should allow for services that understand only the String version of VOQL
- i.e., should not mandate that a TAP service understand the XML version of VOQL (although
- Should include a GET interface to it
- Without impeding Web interfaces to it

- But keep an eye on:
  - How to handle long running queries
  - What to do with the results (where do they go?)
  - How to support distributed querying

Unique identifiers

- Problem:
  - Protocols using VOQL will need unique identifiers for model attributes to be able to access them easily
- Current models:
  - Give UTYPE _and_ UCD to identify bi-univocally an attribute
  - This could be overcome by using two where clauses, but is very inconvenient for Protocol definitions
- Work together with DM group to define how to address the problem

Example case:

- Select both observational and laboratory wavelengths from Line-DM VOQL-aware service:
  - With current utype/ucd, something like
    - SELECT ldm:Line.wavelength[LineType=observed], ldm:Line.wavelength[LineType=laboratory]

    Not very good for parsers
  - With “simplified” model for lines, include the “Observed” or “Laboratory” quality within the uhnique bi-univocal identifier:
    - SELECT ldm:Line.wavelength, ldm:Line.observedWavelength

Example output (non bi-univocal identifiers)

- Current:
  - Observed:
    - Utype ldm:Line.wavelength
    - UCD em.wl
    - Type double
    - LineType observed
  - Laboratory
    - Utype ldm:Line.wavelength
    - UCD em.wl
    - Type double
    - LineType laboratory

Example output (bi-univocal identifiers)

- Possibly future (bi-univocal identifiers)
  - Observed:
    - Utype ldm:Line.wavelength
    - UCD em.wl
    - Type double
  - Laboratory:
    - Utype ldm:Line.observedWavelength
    - UCD em.wl
    - Type double