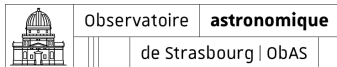


ADQL Next

Grégory Mantelet¹

¹CDS (Centre de Données astronomiques de Strasbourg)

9th May 2023



□ Table of Contents

Features and evolutions

Priorities

Next version

Participation



□ Features and evolutions

Features and evolutions

PEG Grammar

OFFSET by default

CASE structure

Boolean support

Hexa. value + bitwise op.

Array/Vector support

Big numerics

MOC support

INTERVAL support

Query by timestamp

Unit conversion

□ PEG Grammar

- Currently, ADQL provides a BNF Grammar
 - but not machine readable => not possible to validate
 - limitations in the syntax
 - PEG parsers in a lot of languages:
<https://bford.info/packrat/> (e.g. *PEG.js*)
- History:
 - DAL Email: G. Mantelet - 2017-Apr-18
 - Walter Landry slides (Interop May 17, Shanghai)
 - Jon Juaristi slides (Interop May 19, Paris)
 - Jon Juaristi grammar on GitHub repo



□ OFFSET by default

- Proposed by Pierre L.
- Make the OFFSET keyword not optional anymore.



□ CASE structure

- Proposition from Alberto M. - #42 in VOLLT
- Example:

```
SELECT TOP 10  
    CASE WHEN calib_level = 2 THEN 'default' ELSE 'higher'  
FROM ivoa.ObsCore
```



□ Boolean support

- 1st attempt in ADQL-2.1 - #32
- Grammar complexity ; may be simpler with PEG



□ Hexa. value + bitwise op.

- 1st attemp in ADQL-2.1 - #15
- Hexadecimal value (e.g. 0x42)
- Functions or operators (e.g. &, |, ~, <<, >>)?
- **WARNING:** Operation priorities differences from one language to another, signed/unsigned value, ...
 - See [G. Mantelet talk in Oct. Interop 2019 \(Groningen\)](#)



□ Array/Vector support

- More and more needed because of surveys like Gaia
- SELECT arrays
 - formatted as specified by VOTable
 - *should there be something in DALI too?*
- WHERE with arrays:
 - [] operator?
 - functions?
- 1 talk on this topic: see [Tamara presentation](#)
- 1 previous [talk from Markus D.](#) in Oct. 2017



□ Big numerics I

- Question asked by Sara N.
- `BigDecimal` and `BigInteger` in Java
 - Unscaled precision for `BigDecimal`
 - Unlimited sequence of digits for `BigInteger`
 - High precision in storage but also in math. operations
- VOLLT mapping for PostgreSQL:
 - `BigInteger` -> `BIGINT`
 - `BigDecimal` -> `DOUBLE PRECISION`
 - But loosing precision, if very high or precise values



□ Big numerics II

- Possible solutions with PostgreSQL:
 1. BIGINT (*if the BigDecimal value is lower than the max. Long value*)
 2. BINARY (*BigInteger and BigDecimal are stored as bit strings*)
 3. VARCHAR
- *Should there be anything about these special types in DALI and VOTable?*



□ MOC support

- Issue open by Dave M. - #9
- Grégory Mantelet slides (Interop May 17, Shanghai)



□ INTERVAL support.

- Interval/Range of numeric values
- Issue from Pat D. still open - #44
- Terminology issue: `interval` is generally for time whereas `range` is for numerical value
 - but now stuck with the term defined in DALI
- `CAST(...)` can be used to create intervals
 - but a constructor function is also possible
- `INTERSECTS(...)` can be used to set a constraint on a range
 - why not using `BETWEEN` with a different syntax: `mag BETWEEN mag_ranges?`



□ Query by timestamp

- Creation already possible with `CAST('...' AS TIMESTAMP)`
- But no constraint possible
- Suggestions:
 - Equality with `=` and `!= (<>)`
 - Comparison with `<`, `>`, ...
 - Or functions?
 - Or `BETWEEN ... AND ...?`



□ Unit annotation

- Annotating literal values add unit metadata in query result
- Example:

```
SELECT 42m AS "random_value_with_unit", ...
```

- Also useful for math. operation
 - implicit conversion
 - unit inference



□ More string functions

- SUBSTRING(...)
 - Why not also LEFT() and RIGHT()?
- SPLIT_PART(...) (*as defined in PostgreSQL*)



□ Geometries

- Suggested by Alberto M.
1. Add a function to compute the polygonal intersection of 2 given regions - #52
 2. DISTANCE extension between two geometries other than just POINTs
 3. Complex regions defined with intersections and unions thanks to REGION(...)
 4. See OGC standard (OpenGIS® Implementation Standard for Geographic information) for ADQL evolution



□ Priorities

Features and evolutions

Priorities

Next version

Participation



□ Priorities for v2.2

1. PEG Grammar (*and validators*)
2. Better support of INTERVAL and TIMESTAMP (*creation and query*)
3. MOC support (*creation, query and select*) (**collaboration with TAP and DALI**)
4. Array support (**collaboration with DALI**)
5. SUBSTRING(...) and other strings functions
6. CASE structure
7. *OFFSET by default?*



□ Priorities for v2.3

All the other topics?



□ Next version

Features and evolutions

Priorities

Next version

Participation



□ Next version

- Because of PEG Grammar, **v2.2 or v3.0** ?
 - *No major change, so go for v2.2!*



□ Participation

Features and evolutions

Priorities

Next version

Participation



□ Participation

- Create an issue in the GitHub repository:
<https://github.com/ivoa-std/ADQL/issues>
- Start a discussion in Slack: [#adql](#) channel
- . . . or by email: dal@ivoa.net
- Or if too shy, send me directly an email:
gregory.mantelet@astro.unistra.fr

