ADQL-2.1 Status

Grégory Mantelet[1]

[1]CDS (Centre de Données astronomiques de Strasbourg)

19th November 2020

Table of Contents

1. ADQL-2.1 Status

1. ADQL-2.1 Status

- ADQL-2.1 is in **PR since January 2018** *(1st working draft: May 2016)*
- **Goal:** to go RFC (then REC) once the last remaining issues are fixed
- **Status:** very few issues left

2. What is done

2.a. Boolean type 1/3 - #32

- **Goal:**
  - Avoid the =1 in front of functions returning a boolean value
  - And especially for the xmatch syntax recommended in ADQL-2.0:

```
FROM t1 JOIN t2
  ON 1 = CONTAINS(POINT('ICRS', ra, dec),
                  CIRCLE('ICRS', 3., 14., 1/60.))
```

2.a. Boolean type 2/3 - #32

- **But...**
  - The xmatch syntax recommended in ADQL-2.1 is different:

```
FROM t1 JOIN t2
  ON DISTANCE(POINT(ra, dec), POINT(3., 14.))
     < 1/60.
```

  - Grammar complexity
    - allow `=1` and `=True` in the same time
    - complex grammar when no `=1/True/0/False` is used

2.a. Boolean type 3/3 - #32

- **Status:** Postponed to a next version of ADQL *(waiting for reviews in GitHub)*
- **Bonus:** in the next version of ADQL grammar changes easier thanks to the migration to PEG grammar.

2.b. Geometries arg. - #29

- **Problem:**
  - The coord. sys. argument is also now optional
  - ADQL-2.1 allows a 2nd signature for `CIRCLE` and `POLYGON`:

```
CIRCLE(  12   ,  23   , 1/60.) -- ADQL-2.0
CIRCLE(POINT(12, 23), 1/60.) -- ADQL-2.1+
CIRCLE(   my_point   , 1/60.) -- ADQL-2.1+
```

  - OK for `CIRCLE`, **but not** for `POLYGON`

Example with POLYGON

```
POLYGON(colCooSys,
        colRa1, colDec1,
        colRa2, colDec2,
        colRa3, colDec3)
POLYGON(colRa1, colDec1,
        colRa2, colDec2,
        colRa3, colDec3)
POLYGON(colPoint1,
        colPoint2,
        colPoint3)
-- but also:
POLYGON(colPoint1,
        colPoint2, colPoint3,
        colPoint4, colPoint5,
        colPoint6, colPoint7)
-- which is ambiguous with the example with colCooSys !!!!
```

2.b. Geometries arg. - #29

- **Status:** solved with the Erratum 3

2.c. UDF catalogue

Proposed Endorsed Note - Catalogue of ADQL User Defined Functions, *Jon Juaristi Campillo, et al.*

- **Goal:** to gather frequent and common UDF definitions so that encouraging ADQL implementers to adopt them as much as possible when needed.
- **Status:** Currently in RFC
- **Once Endorsed Note**, it will solve 2 ADQL-2.1 issues:
  - adding a note to the UDF Catalogue in the ADQL-2.1 document - *#16*
  - coordinate system management - *#10*

3. Work in progress

3.a. Set operators 1/4 - #29

- **Problems:**
  - fix description of `INTERSECT` *(which was the same as `EXCEPT`)*
  - operators precedence issue *(`INTERSECT` and operations between parenthesis must be resolved first)*

3.a. Set operators 2/4 - #29

- **Status:**
  - correction proposal already published *(see PullRequest#42)*
    - **Examples of allowed set operations in ADQL-2.1:**

```
    SELECT hip FROM hipparcos WHERE hip <= 10
UNION
    (SELECT TOP 10 hip FROM hipparcos WHERE hip > 10 ORDER BY 1)
ORDER BY 1 DESC
```

```
(
    SELECT hip FROM hipparcos WHERE hip <= 10
UNION
    SELECT hip FROM hipparcos WHERE hip > 120406
)
INTERSECT
    SELECT hip FROM hipparcos WHERE (hip <= 10 OR hip > 120406) AND mod(hip
= 0
ORDER BY 1 DESC
```

[. . .] *(see more on GitHub)*

3.a. Set operators 3/4 - #29

- **Status:**
  - correction proposal already published *(see PullRequest#42)*
  - finishing implementation tests with PostgreSQL, MySQL and SQLServer
    - Translation into SQL a bit tricky for `INTERSECT` and `EXCEPT` with MySQL, but possible:

ADQL:

```
    SELECT hip FROM hipparcos WHERE hip <= 10
INTERSECT
    SELECT hip FROM hipparcos WHERE hip <= 10 AND mod(hip, 2) = 0
ORDER BY 1 DESC;
```

MySQL:

```
SELECT DISTINCT t1.*
FROM         (SELECT hip FROM hipparcos WHERE hip <= 10) AS t1
  INNER JOIN (SELECT hip FROM hipparcos WHERE hip <= 10 AND mod(hip, 2) = 0) AS
t2
    ON t1.hip = t2.hip
ORDER BY 1 DESC;
```

3.a. Set operators 4/4 - #29

- **Status:**
  - correction proposal already published *(see PullRequest#42)*
  - finishing implementation tests with PostgreSQL, MySQL and SQLServer
    - Translation into SQL a bit tricky for `INTERSECT` and `EXCEPT` with MySQL, but possible:
    - embedded CTEs not allowed in SQLServer

```
    SELECT hip FROM hipparcos WHERE hip <= 10
UNION
    (
        WITH tt AS (SELECT hip FROM hipparcos WHERE hip > 120406)
        SELECT hip FROM tt
    )
ORDER BY 1 DESC;
```

=> **Solution:** in ADQL, allow CTEs only at the root level

4. To do

4. To do

- **CAST vs Constructors** - *#11*

CAST	Constructor
`CAST('2020-05-07', TIMESTAMP)`	`TIMESTAMP('2020-05-07')`
`CAST(30, SMALLINT)`	`SMALLINT(30)`
`CAST('23 42', POINT)`	`POINT(23, 42)`

- **Intervals overlap** - *#44*
  - *If no difficulties at all: ADQL-2.1*
  - *Otherwise: next ADQL version*
- **Adding `UPPER(...)`**
  - *`ILIKE` and `LOWER(...)` now exist in 2.1*

5. ADQL Implementations

5. ADQL Implementations

- **CADC** : ADQL-2.0
- **DACHS** : ADQL-2.1 *(to be checked)*
- **VOLLT/ADQL-Lib** : ADQL-2.1 *(to be checked)*

Conclusion

Let's keep in touch!

- GitHub project *(Issues, PullRequests, and Projects overview)*
- dal@ivoa.net *(email)*
- #adql channel on SLACK *(chat)*

Appendices

A1. ADQL-2.1 in brief...

Feature	Status
`UNION` / `INTERSECT` / `EXCEPT`	✎
`TIMESTAMP(...)` + type constructors or `CAST(...)`	▷
Interval functions (intersects/overlaps)	▷
Optional features	✓
`WITH` (or Common Table Expression = CTE)	✓
`OFFSET`	✓
Recommended x-match syntax (now `DISTANCE(...)`)	✓
Grouping/Ordering by an expression	✓
`ILIKE`	✓
`LOWER(...)`	✓
`IN_UNIT(...)`	✓
Fix `MOD(...)`, `RAND(...)`, `ROUND(...)` and `TRUNCATE(...)`	✓
Deprecation of `BOX`	✓
Geometry constructor with "exploded" points	✓
Deprecation of coord. sys. argument + `COORDSYS(...)`	✓
UDF Catalogue	✓
Clarifications on datatypes	✓
BLOB and CLOB	✓
Hexadecimal values/operations	L
Bitwise operators	L
Boolean datatype	L

A2. After ADQL-2.1...

- PEG grammar
- Arrays
- Interval *(see DALI)*
- MOC
- Boolean data-type
- Hexadecimal notation + Bitwise operations
- Unit annotation
- Substring function *(seems to already exist in SQL-92)*
- Geometries:
  - *see OGC standard (OpenGIS® Implementation Standard for Geographic information) for ADQL evolution*
  - `DISTANCE` between other geometries than `POINT`
  - complex regions defined with intersections