

Wrapping up ADQL-2.1

Grégory Mantelet¹

¹CDS (Centre de Données astronomiques de Strasbourg)

26th May 2021



Université
de Strasbourg

□ Table of Contents

1. ADQL-2.1 Status

2. What's new?

3. Implementations

4. What's next?

Conclusion

□ 1. ADQL-2.1 Status

- **Status:** no issue left for 2.1
- New PR coming very soon
- Then start RFC


□ ADQL 2.0

```
SELECT
  fav.name,
  survey.*
FROM
  my_favorite_stars      AS fav
JOIN best_survey_ever AS survey
  ON 1=CONTAINS( POINT('ICRS', fav.ra, fav.dec),
                CIRCLE('ICRS', survey.ra, survey.dec, 1./360.))
```

Classical ADQL-2.0 query with a positional cross-match

ADQL 2.1


```
SELECT
  fav.name,
  survey.*
FROM
  my_favorite_stars AS fav
JOIN best_survey_ever AS survey
  ON 1=CONTAINS( POINT(fav.ra, fav.dec),
                CIRCLE(survey.ra, survey.dec, 1./360.))
```

 **Coordinate system argument deprecated**
→ optional


Apply to:

- POINT(...)
- CIRCLE(...)
- BOX(...)
- POLYGON(...)

sec. 4.2.4, p.25

 **COORD_SYS(...)**
deprecated

*sec. 4.2.6, p.26
sec. 4.2.15, p.36*

 **BOX(...)**
deprecated

Instead, in function of what BOX means to you, use:

- WHERE ra BETWEEN ... AND ...
AND dec BETWEEN ... AND ...
- POLYGON(...)

sec. 4.2.9, p.29-30

ADQL 2.1

```
SELECT
  fav.name,
  survey.*
FROM
  my_favorite_stars AS fav
JOIN best_survey_ever AS survey
  ON 1=CONTAINS(fav.position, CIRCLE(survey.position, 1./360.))
```



Give a coordinates pair
instead of a POINT (...)

Apply to:

- DISTANCE(...)
- CIRCLE(...)
- BOX(...)
- POLYGON(...)

Both syntax are possible.

sec. 4.2.6, p.26
sec. 4.2.9, p.29-31
sec. 4.2.11, p.31-32
sec. 4.2.16, p.36-37
sec. 4.2.19, p.40-42

□ ADQL 2.1

```
SELECT
  fav.name,
  survey.*
FROM
  my_favorite_stars      AS fav
JOIN best_survey_ever AS survey
  ON DISTANCE(fav.position, survey.position) < 1./360.
```



Preferred positional cross-match syntax

Clients proposing crossmatch-like queries are advised to phrase them this way rather than semantically equivalent alternatives, and services are encouraged to ensure that this form of join is executed efficiently ; this might involve identifying such ADQL input clauses and rewriting them appropriately for efficient processing on the database backend.

sec. 4.2.7, p.27-28

□ ADQL 2.1

```
SELECT
  fav.name,
  survey.*
FROM
  my_favorite_stars      AS fav
  JOIN best_survey_ever AS survey
  ON DISTANCE(fav.ra, fav.dec, survey.ra, survey.dec) < 1./360.
```



Preferred positional cross-match syntax

Clients proposing crossmatch-like queries are advised to phrase them this way rather than semantically equivalent alternatives, and services are encouraged to ensure that this form of join is executed efficiently; this might involve identifying such ADQL input clauses and rewriting them appropriately for efficient processing on the database backend.

sec. 4.2.7, p.27-28

ADQL 2.1

```
SELECT
  fav.name,
  survey.id           AS survey_id,
  LOWER(survey.star_cat) AS category
FROM
  my_favorite_stars  AS fav
  JOIN best_survey_ever AS survey
  ON DISTANCE(fav.position, survey.position) < 1./360.
WHERE survey.star_cat ILIKE '%star%'
```



String functions

- LOWER(...)
- UPPER(...)
- ILIKE

sec. 4.4.1-3, p.43-44

ADQL 2.1

```
SELECT
  fav.name                AS star_name,
  survey.id              AS survey_id,
  LOWER(survey.star_cat) AS category,
  CAST(survey.iso8601_time AS TIMESTAMP) AS "when"
FROM
  my_favorite_stars      AS fav
  JOIN best_survey_ever AS survey
    ON DISTANCE(fav.position, survey.position) < 1./360.
WHERE survey.star_cat ILIKE '%star%'
```



CAST(... AS ...)

Minimal set of supported target types:

- INTEGER, SMALLINT, BIGINT
- REAL, DOUBLE PRECISION
- CHAR, VARCHAR
- TIMESTAMP

sec. 4.7, p.50-53



Type system

ADQL-2.1 defines a type system including a lot of types.

Not all of them can be (yet) manipulated or created by standard functions or operators.

Examples: TIMESTAMP, INTERVAL, BLOB, ...

sec. 3, p.17-22



Identifiers

Be careful when naming columns, tables, schema and aliases!

See sec. 2.1.2-7 (p.9-10) about correct identifiers, reserved keywords and case sensitivity.

Especially:

Data providers should avoid defining column names using delimited identifiers.

Sec. 2.1.7, p.10

ADQL 2.1

```
SELECT
  fav.name                AS star_name,
  survey.id              AS survey_id,
  LOWER(survey.star_cat) AS category,
  ivo_healpix_index[survey.ra, survey.dec, fav.hpx_order] AS hpx_index,
  gavo_to_mjd[CAST(survey.iso8601_time AS TIMESTAMP)] AS "when"
FROM
  my_favorite_stars      AS fav
  JOIN best_survey_ever AS survey
    ON DISTANCE(fav.position, survey.position) < 1./360.
WHERE survey.star_cat ILIKE '%star%'
```

User Defined Functions (UDF)

In order to avoid name conflicts, user defined function names **SHOULD include a prefix** which indicates the name of the institute or project which created the function.

sec. 4.3.1, p.43

The **ivo** prefix is reserved for functions that have been defined in an IVOA specification.

sec. 4.3.1, p.43


In order to avoid different signatures for the same functions, it is recommended that ADQL implementers follow as much as possible the User Defined Function signatures listed in the **Catalogue of ADQL User Defined Functions**

(Campillo, J. J. and Demleitner, M. (2021)).

sec. 4.3.1, p.43


□ ADQL 2.1

```
SELECT DISTINCT
  ivo_healpix_index(survey.ra, survey.dec, fav.hpx_order) AS hpx_index
FROM
  my_favorite_stars      AS fav
  JOIN best_survey_ever AS survey
    ON DISTANCE(fav.position, survey.position) < 1./360.
WHERE survey.star_cat ILIKE '%star%'
```

 A step forward MOC...

ADQL 2.1

```
SELECT
  ivo_healpix_index(survey.ra, survey.dec, fav.hpx_order) AS hpx_index,
  COUNT(*) AS cnt
FROM
  my_favorite_stars AS fav
  JOIN best_survey_ever AS survey
    ON DISTANCE(fav.position, survey.position) < 1./360.
WHERE survey.star_cat ILIKE '%star%'
GROUP BY hpx_index
```

 A step forward Healpix Map...

ADQL 2.1

```
SELECT TOP 1000
  fav.name                AS star_name,
  survey.id               AS survey_id,
  LOWER(survey.star_cat) AS category,
  ivo_healpix_index(survey.ra, survey.dec, fav.hpx_order) AS hpx_index,
  gavo_to_mjd(CAST(survey.iso8601_time AS TIMESTAMP)) AS "when"
FROM
  my_favorite_stars AS fav
  JOIN best_survey_ever AS survey
    ON DISTANCE(fav.position, survey.position) < 1./360.
WHERE survey.star_cat ILIKE '%star%'
ORDER BY survey.id
OFFSET 1000
```



OFFSET ("pagination")

Evaluation order:

1. ORDER BY
2. OFFSET
3. TOP

sec. 4.9.1, p.55

ADQL 2.1

```
SELECT TOP 1000
  fav.name                AS star_name,
  survey.id              AS survey_id,
  LOWER(survey.star_cat) AS category,
  ivo_healpix_index(survey.ra, survey.dec, fav.hpx_order) AS hpx_index,
  gavo_to_mjd(CAST(survey.iso8601_time AS TIMESTAMP)) AS "when",
  IN_UNIT(survey.wavelength, 'Angstrom') AS wavelength
FROM
  my_favorite_stars AS fav
  JOIN best_survey_ever AS survey
  ON DISTANCE(fav.position, survey.position) < 1./360.
WHERE survey.star_cat ILIKE '%star%'
ORDER BY survey.id
OFFSET 1000
```



IN_UNIT(...)

The 2nd argument (the target unit) MUST be a valid VOUnit ([Derriere and Gray et al. \(2014\)](#)).

A unitless value can not be converted. A value with a unit can not be converted to a unitless one.

Error reported in case the unit is invalid or if the conversion is not possible.

sec. 4.8.1, p.53



Implementation

Implementation rather difficult (only DACHS does currently).

An advanced unit inference mechanism may also be implemented in order to resolve complex expressions.

ADQL 2.1

```
SELECT survey.id
FROM my_favorite_stars AS fav
JOIN best_survey_ever AS survey
  ON DISTANCE(fav.position, survey.position) < 1./360.
WHERE survey.star_cat ILIKE '%star%'
UNION
SELECT survey.id
FROM my_colleague_stars AS coll
JOIN best_survey_ever AS survey
  ON DISTANCE(coll.position, survey.position) < 1./360.
WHERE survey.star_cat ILIKE '%star%'
ORDER BY 1
```



Set Operations UNION, INTERSECT, EXCEPT

UNION/INTERSECT/EXCEPT ALL preserve duplicates.

The following clauses apply to the result of the set operation:

- WITH (*see next slide*)
- ORDER BY
- GROUP BY
- OFFSET

ORDER/GROUP BY and OFFSET can be applied to an individual query if this one is wrapped between parentheses.

Priority rules:

1. Set operations between parentheses
2. INTERSECT
3. UNION and EXCEPT (*from left to right*)

sec. 4.6, p.47-50

ADQL 2.1

```
WITH our_stars AS (  
  SELECT name, POINT(ra, dec) AS position FROM my_favorite_stars  
  UNION  
  SELECT name, POINT(ra, dec) AS position FROM my_colleague_stars  
)  
SELECT TOP 1000  
  fav.name                               AS star_name,  
  survey.id                             AS survey_id,  
  LOWER(survey.star_cat)                 AS category,  
  ivo_healpix_index(survey.ra, survey.dec, 10) AS hpx_index,  
  gavo_to_mjd(CAST(survey.iso8601_time AS TIMESTAMP)) AS "when",  
  IN_UNIT(survey.wavelength, 'Angstrom') AS wavelength  
FROM   our_stars AS fav  
  JOIN best_survey_ever AS survey  
  ON DISTANCE(fav.position, survey.position) < 1./360.  
WHERE survey.star_cat ILIKE '%star%'  
ORDER BY survey_id  
OFFSET 1000
```



Common Table Expressions (CTE) WITH

The WITH operator lets create one or more temporary named result sets that can be referred to elsewhere in the query.

Not supported:

- Recursive CTE
- In sub-queries

sec. 4.5, p.45-46

□ 3. Implementations

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

□ 4. What's next?

- PEG grammar
- Query by timestamp
- Create and query Interval (*see DALI*)
- MOC
- Arrays
- 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

□ Suggestions, issues, ... ?

- Create an issue in the GitHub repository:
<https://github.com/ivoa-std/ADQL>
- 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

□ Thanks. . .

Special thanks to Dave Morris for his tremendous work on most of ADQL-2.1 features ;-)

And of course, thanks to all other contributors for their help and reviews ;-)