A proposal for vector math in ADQL

Jon Juaristi Campillo\textsuperscript{1,2} Markus Demleitner\textsuperscript{1,2}

\textsuperscript{1}Astronomisches Rechen-Institut (ARI), Zentrum für Astronomie (ZAH), Universität Heidelberg

\textsuperscript{2}German Astrophysical Virtual Observatory (GAVO)

October 2022 IVOA Interoperability Meeting
DAL session – 2022-10-18
Contents

1 Rationale

2 Proposal

3 Implementation notes

4 Current status
Why vector math?

- Tables containing massive amounts of vector are becoming commonplace (e.g., Gaia DR3 low resolution spectra).
- No toolset to do server-side arrays, which might be desirable to the end user – by leveraging computations.
- Able to enhance ADQL as a tool for server-side analyses.
Element access

- By using [element-index], it being an integer value expression
- 1-based arrays – rather than 0-based
- Elements outside the [1, n] range = NULL

Example: my_array = {0.4, 4.2, 8.1}

my_array[0] NULL
my_array[1] 0.4
my_array[4] NULL
Basic math

- Component-wise operations:
  - Sum: \( vec1 + vec2 \)
  - Subtraction: \( vec1 - vec2 \)
  - Multiplication: \( vec1 \ast vec2 \)
  - Division: \( vec1/vec2 \)
    (whenever vectors have unequal lengths, the result is padded with NaNs to the length of the longer one)

- Scalar multiplications: \( scalar \ast vec, vec \ast scalar \)

- **Floating point** scalar division: \( vec/scal = (1/scalar) \ast vec \)
arr_dot(vec1, vec2)

- Scalar product of two vectors
- When lengths are unequal, the short vector is padded with NaNs to the length of the longer vector — the scalar product of vectors of unequal length is NaN
Array aggregation

- `arr_avg(arr)` arithmetic mean of elements
- `arr_max(arr)` largest element
- `arr_min(arr)` smallest element
- `arr_sum(arr)` sum of all elements

- Work like SQL aggregate functions (on the elements of arrays).
Aggregate functions for arrays

- AVG, MIN, SUM, and MAX work component-wise.
- Undefined result (by now) when computing aggregates over arrays of different length, options being:
  - Returning an error
  - Extend with NaN
  - Extend with NULL – what Postgres (used by DaCHS) does.
Array map

arr_map(expr_over_x, arr)

- Computes a new array by binding each element of arr to x in turn and then computing expr_over_x
- expr_over_x: ADQL numeric_value_expression which can use that can use column references as usual (except the reserved name x).

Example: arr_map(power(10, x), m) → [power(10, m[1]), power(10, m[2]), power(10, m[3])...]

Array support on relational database management systems is relatively scarce, with a few exceptions...

...such as **PostgreSQL**, on which this proposal has been **implemented** – as it also supports multidimensional data.
Element access

- Two different ways:
  - Single element: identifier be succeeded by [index]
  - Multiple elements (as a sub-array): identifier succeeded by [lower-bound:upper-bound]
- Adding these features as-is would imply some changes in the ADQL grammar
- Or as functions: e.g., array_item(index) and/or array_slice(lower, upper)
Wherever possible, mathematical operations which are compatible with arrays have been overloaded.

- It is a sort of syntactic sugar to avoid proliferation of explicit functions.
- They take – alongside the function –, two operands and a commutator (as a minimum).
- Unavoidable duplication in the case of scalar multiplication and division: to ensure commutation.
(User Defined) Functions

- The functionality not covered by operations (such as aggregate functions) are available as user defined functions.
- No ivo_ prefix currently: not an extension but part of ADQL.
- There could be a possibility to add some of them into the standard in the near future – we’re hoping for ADQL 2.2.
For the time being, vector support can be declared by using a temporary feature type with a g-vo authority:
(it would eventually become an IVOA ivoid)

```xml
<languageFeatures
    type="ivo://org.gavo.dc/std/exts#extra-adql-keywords">  
    <feature>
        <form>VECTORMATH</form>
        <description>
            You can compute with vectors here. See
            https://wiki.ivoa.net/twiki/bin/view/IVOA/ADQLVectorMath
            for an overview of the functions and operators available.
        </description>
    </feature>
</languageFeatures>
```
Adoption

- As of now, the data provider which “fully” supports vector math is GAVO.
- “Slicing” arrays is the only feature to be implemented.
- A writeup of a real world example by Markus Demleitner is available at the GAVO blog.
Any questions?

If you would like to contact us, feel free to send an email:
- juaristi@uni-heidelberg.de (Jon)
- msdemlei@ari.uni-heidelberg.de (Markus)