GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

# VO at the Limit: Optionality Considered Harmful

Markus Demleitner

IVOA Southern Spring Interop 2024, Malta Nov 14-17
Apps WG

## The Story

For GAVO's Big VO Course (Demleitner and Heinl et al., 2024), I wanted to write an all-VO TAP query:

"Give me all measurements of proper motions in the vicinity of point X."

This turned out to be a surprisingly bad pain.

Here is the story.

## Find Tables With pos.pm Columns

The pyVO registry API cannot return information on tables yet. Thus, we run a custom RegTAP query:

```
SELECT DISTINCT access_url, table_name
FROM rr.interface
NATURAL JOIN rr.capability
NATURAL JOIN rr.res_table
NATURAL JOIN rr.table_column
NATURAL JOIN rr.stc_spatial
WHERE
    standard_id LIKE 'ivo://ivoa.net/std/tap%'
    AND ucd LIKE 'pos.pm%'
    AND 1=INTERSECTS(POINT({RA}, {DEC}, {SR}), coverage)
    AND (table_type!='output' OR table_type IS NULL)
```

## Query Construction: UCDs

It would be nice if astropy tables had a `fieldname_with_ucd` function. What I did instead was not *terribly* hard, though:

```
def fieldname_with_ucd(ucd, table):
    ucd = ucd.lower()
    for col in table.columns:
        if col.ucd and col.ucd.lower()==ucd:
            return col.name
    raise KeyError(ucd)
```

The UCD's case insensitivity has doubled the complexity of this function.

Mitigation: no case folding any more (in new-ish machine interfaces).

## Query Construction: Delimited Identifiers

```
for dest_name, ucd, unit, type in RESULT_SCHEMA:
    select_clause.append("{} AS {}".format(
        fieldname_with_ucd(ucd, db_table),
        dest_name))
```

This fails quickly because of a breach of VODataService:

```
Incorrect ADQL query:
Encountered "/". Was expecting one of: <EOF> "." "," ";" "AS"
 "WHERE" "GROUP" "HAVING" "ORDER" "\""
 <REGULAR_IDENTIFIER_CANDIDATE> "NATURAL" "INNER" "LEFT"
 "RIGHT" "FULL" "JOIN"
```

> *[W]hen delimited identifiers are used for column names on the relational side [. . . ] the quotes must be part of name's value, and the capitalisation used in the DDL must be preserved.*

## Saving Requests: Doing UNION

When you want to query potentially thousands of tables, it would be cool to run a server-side UNION over the queries.

But UNION is optional. I need hedging code:

```
knows_union = svc.get_tap_capability().get_adql().get_feature(
    "ivo://ivoa.net/std/TAPRegExt#features-adql-sets", "UNION")

def feed_rows(astropy_table):
    for row in astropy_table:
        result_rows.append(dict(zip(row.colnames, row.as_void())))

if knows_union:
    feed_rows(svc.run_sync(
        " UNION ".join(queries)).to_table())
else:
    for query in queries:
        feed_rows(svc.run_sync(query).to_table())
```

## But It Works, Doesn't It?

Sure, you *can* write code like that. But:

- Complicated sensing (`get_tap_capability().get_adql().get_feature` oh my)
- duplicate code
- complicated injection of common code (the inner `feed_rows` function)

Mitigation: require as much as we can; we *can* (in general) make new requirements in minor versions. We just cannot drop them.

And talk to data publishers so new standards (in this case, ADQL 2.1) propagate faster.

## Unit Coversion

We want a common result schema, *including units*.

It would be great if we could just say

```
SELECT IN_UNIT(<pmra-column>, 'mas/yr') as pmra...
```

– but `IN_UNIT` is optional, too, so again: Multiple code paths.

At least *this* is not unreasonably difficult to do on the client side.

## Casting

When you do UNIONs, you quickly run into errors like these:

```
pyvo.dal.exceptions.DALQueryError: Field query: UNION types integer
and text cannot be matched LINE 1: ...S(12), RADIANS(13)), RADIANS(0.1))))
UNION SELECT localid AS...
```

This is because row identifiers sometimes are strings, somtimes integers. This would be easily fixed like this:

```
SELECT CAST(<identifier-column> AS TEXT) AS identifier
```

Except: CAST is optional.

## Another Feature Switch

```
knows_cast = svc.get_tap_capability().get_adql().get_feature(
            "ivo://ivoa.net/std/TAPRegExt#features-adql-type", "CAST")
for dest_name, ucd, unit, type in RESULT_SCHEMA:
   if type and knows_cast:
       select_clause.append("CAST({} AS {}) AS {}".format(
           perhaps_quote(fieldname_with_ucd(ucd, db_table)),
           type,
           dest_name))
   else:
       # Don't cast and hope for the best
       select_clause.append("{} AS {}".format(
           perhaps_quote(fieldname_with_ucd(ucd, db_table)),
           dest_name))
```

(ok: in practice services that know UNION know about CAST, too).

## Conclusion

- Rather have fewer features than optional ones
- Don’t be case-insensitive
- Have task forces talking to data providers, advising them on how to upgrade. Let’s have less of ancient versions complicating clients forever. P3T, do you hear me?
- Be a bit faster to fix non-compliant services
- Feature discovery is nice. Not having to do it is nicer

## See Also

Demleitner, M., Heinl, H. and Wambsganss, J. (2024), ‘Using the virtual observatory’, Lecture notes for a course given at Universität Heidelberg, summer semester 2024. doi:10.21938/avVAxDlGOiu0Byv7NOZCsQ, https://docs.g-vo.org/vocourse.