DAL Running Meeting #15 - ADQL Validator

Date: *Fri. 10 March 2023 - 6am UTC*
Participants:

- Grégory Mantelet (CDS)
- ...

Lyonetia repository

- **Initial goal:** validate ADQL implementations ; only two: DACHS and ROE *(almost no tests)*
- **Goal of the ADQL validator:** strictly check that the official standard ADQL grammar allows features and use cases described in the ADQL standard.

Validator

- Written in Java using VOLLT/ADQL-Lib
- No non-standard extension enabled
- Both ADQL 2.0 and 2.1 can be tested

Errata

- This validation step helped us to bring one issue with ADQL grammar since 2.0.
    - Proposed erratum: Erratum 4: NULL as valid value expression
        - ***Please review it, and comment if needed***
    - ADQL-2.1 grammar and validator already adapted

Unresolved tests

ivoa/1_select.xml#fe956f66-3c3e-410d-baa4-59305da8ddfc

- **Description:**

  > Selecting columns with an incorrect name syntax (not starting with a letter) and no double quotes.

- **Query:**

  ```
  SELECT _weird_name FROM stars
  ```

- **Expected:** Failure
- **Reason:** since 2.0, according to the ADQL grammar, a regular column or function name cannot start with a character different from `[A-Za-z]`
- **Status:** Moderate opposition from Markus who accepts such column name in DACHS. However, Markus is OK to keep this test as it is. I personally want to keep it because it is a strict application of the ADQL-2.x grammar ; the goal of the ADQL validator is to blindy follow the standard grammar.
- **Summary of the discussion in GitHub:**
  - Markus - https://github.com/ivoa/lyonetia/pull/16#issuecomment-1400621607

    > DaCHS, as a little extension, actually accepts names starting with an underscore, and with postgres, that's very convenient. While I'm all for exercising the name parser, perhaps we can use a name broken in some other way, just because leading-underscore names are such an obvious extension people may want to have? Perhaps .somename? I give you that may exercise different rules.

  - Grégory - https://github.com/ivoa/lyonetia/pull/16#issuecomment-1401693892

    > This time, I agree and I do not agree with you. I explain. I personally agree that in DBMS (like Postgres) this is not an issue. So, in ADQL, I would personally allow column names prefixed with an underscore.
    >
    > However, since ADQL-2.0, a column name **must** start with an alphabetic letter, according to the BNF. This is still the case in ADQL-2.1:
    >
    > ```
    > <column_name> ::= <identifier>
    >
    > <identifier> ::= <regular_identifier> | <delimited_identifier>
    >
    > <regular_identifier> ::=
    >                       <simple_Latin_letter>...
    >                       [ { <digit> | <simple_Latin_letter> |
    > <underscore> }... ]
    > ```
    >
    > The same rule applies to a UDF name.
    >
    > Whether or not I personally disagree with this, we should NOT allow our validator to ignore some part of the ADQL standard. Otherwise, we could not call it a "validator".
    >
    > So, two solutions here:

> 1. We now allow column names and UDF to start with an underscore. ADQL-2.1 should then be updated.
> 2. We don't change anything. So, we keep the ADQL BNF as it is currently and we keep this validation query (ivoa/1_select.xml#fe956f66-3c3e-410d-baa4-59305da8ddfc).
>
> What do you think?

- Markus - https://github.com/ivoa/lyonetia/pull/16#issuecomment-1404983875

> Absolutely. And I don't think we should change that, as that may make implementation a lot harder on top of RDMSes that are less forgiving than postgres.
>
> [...]
>
> Well... we have some leeway as to what tests we run and what we don't test against. For instance, DaCHS extends ADQL with TABLESAMPLE. We *could* include a test that an implementation rejects something like
>
> SELECT * from foo TABLESAMPLE(0.1)
>
> which clearly is invalid ADQL 2.1 -- but we don't, as I'd argue we don't want to be in people's ways when they prototype features for the next versions of ADQL.
>
> Similarly, in this case we *could* say identifiers with leading underscores may be a reasonable extension and we simply don't test whether or not an implementation rejects identifiers with leading underscores.
>
> By my criteria what to validate and what not to validate, https://blog.g-vo.org/requirements-and-validators.html, I don't see a compelling argument why we should make sure parsing fails in this case. But I certainly might be missing something.
>
> [...]
>
> If my reasoning above is about right, I'd say we can just drop the test and not exercise the whole thing.
>
> But then if you want to keep the test, I won't complain (and just think about an ignore pattern in my test suite).

ivoa/1_select.xml#85e497ac-4403-4f84-8aa0-a3ee3da514cc

- **Description:**

  > Selecting columns with another incorrect name syntax (starting with a
  >
  > digit) and no double quotes.

- **Query:**

  ```
  SELECT 2weird FROM stars
  ```

- **Expected:** Failure
- **Reason:** since 2.0, according to the ADQL grammar, a regular column or function name cannot start with a character different from `[A-Za-z]`

- **Status:** some DBMS (e.g. Postgres, Oracle, but not MySQL and SQLite) interpret this as `2 AS weird` because the `AS` keyword is optional. Proposition of Markus: forget about this test. My proposition: keep it ; it will save our users sanity by forcing all ADQL implementations to have the same behavior whatever is the DBMS they use.
- **Summary of the discussion in GitHub:**
  - Markus - https://github.com/ivoa/lyonetia/pull/16#issuecomment-1400621607

    > Regrettably, that's valid SQL (and ADQL). What's there is an ugly short form of SELECT 2 AS weird FROM stars.

  - Grégory - https://github.com/ivoa/lyonetia/pull/16#issuecomment-1401693892

    > Should it be really valid in ADQL? If it does, it could lead to unexpected behavior from what expect our users. I agree that it is a good thing to follow as closely as possible SQL, but should we strictly follow it when it is obviously not desired? I would personally save our users sanity by strictly forbidding regular column names prefixed by a digit. A regular ADQL user expects a column name prefixed by a digit, and since this is forbidden in ADQL, he will get an error ; this is, according to me, better than allowing an implicit column name alias.
    >
    > This "issue" actually comes from the fact that the BNF does not specify how grammar tokens should be identified and how space characters should be considered. This is one of the reasons why we want to move to a PEG grammar. So the question is: do we want to allow such dangerous syntax when we will have a PEG grammar? If yes, how would you write that in PEG syntax? (I don't think it will be easy)
    >
    > On the implementation side, I don't think I can easily change the parsing mechanism of my parser to allow such ugly thing ; it could be possible, but the trick will be as ugly as this syntax.

  - Markus - https://github.com/ivoa/lyonetia/pull/16#issuecomment-1404983875

    > True -- it sure surprised me, and I had to hack quite a bit to make DaCHS' parser accept abominations like these.
    >
    > [...]
    >
    > Ok... interesting. I was just staring a bit at section 5.2 of SQL92 trying to figure out SQL's tokenisation rules, since so far I was pretty sure SQL engines are required to parse 2foo as 2 AS foo (and frankly, I deplore that the AS is optional quite a bit as well). Interestingly, this doesn't seem to have much factual basis.
    >
    > And SQL92 in 5.2 says: Any may be followed by a . A shall be followed by a or a . If the Format does not allow a to be followed by a , then that   shall be followed by a .
    >
    > And both and are -s. I *think* the cited language forbids two nondelimiting tokens next to each other. Empirically, Postgres understands 2foo, but sqlite3 doesn't. That made me try the SQL Fiddle. MySQL doesn't, Oracle does, SQLServer is currently broken there.
    >
    > So... yeah, let's consider it outlawed. People devious enough to write such a thing deserve no better, and they won't have much fun when they directly talk to SQL engines anyway.

  - Markus - https://github.com/ivoa/lyonetia/pull/16#issuecomment-1405008346

    > Well, it turns out it's not quite as simple. Looking at my grammar (and I don't think this will be much different for some other PEG grammar), it actually seems to be quite involved to disallow 2foo and at the same time accept (2)foo.
    >
    > The straighforward way to write the rules would be something like
    >
    > ::= (AS |)
    >
    > ::= []
    >
    > but that outlaws the (2)foo, which I'd rather not.   Having different rules depending on whether ends with a is equally unattractive.
    >
    > Perhaps I start liking 2foo after all. Or we require the AS, which would immediately solve the whole problem (but would probably break many queries that are out there).

ivoa/0_whitespace#a66c56e6-f18d-11e8-96df-28d244962af0

- **Description:**

  > Order by internal whitespace and stuck sort direction

- **Query:**

```
select * from bar order   by foo asc, 2desc
```

- **Expected:** Success
- **Status:** Currently failing in the ADQL validator because of the missing space between `2` and `desc`.

ivoa/0_whitespace.xml#d4c3a72c-4458-11e6-96e2-28b2bdcff70b, #dbef92f4-4458-11e6-96e2-28b2bdcff70b, #1f5d27dc-450c-11e6-8564-332de33a5c03

- **Queries:**

```
-- d4c3a72c-4458-11e6-96e2-28b2bdcff70b
select x from t1 INNER JOIN (t2 JOIN t3)
-- dbef92f4-4458-11e6-96e2-28b2bdcff70b
select x from (t1 JOIN t4) FULL OUTER JOIN (t2 JOIN t3)
-- 1f5d27dc-450c-11e6-8564-332de33a5c03
select * from ( select * from urks.a, b, ( select * from c, monk.d ) as q )
as r join ( select * from x, y ) as p
```

- **Expected:** Success
- **Status:** All of them actually fail in a database because neither `NATURAL` nor a join condition is provided. Consequently, we decided to change their expected result into `Failure` instead. However, strictly speaking, they are allowed by both the ADQL-2.x and SQL-92 grammars, though no existing DBMS would run them successfully. Then, we could adapt the ADQL grammar in order to require a join condition or to use `NATURAL`, but as SQL-92 does allow

this flexibility and that all ADQL implementations would fail anyway, it has been considered simpler to not change anything.

- *But maybe there should be a note in the ADQL prose stating that a non `NATURAL JOIN` must always have a join condition.*

- **Summary of the discussion in GitHub:**
  - Gregory - https://github.com/ivoa/lyonetia/pull/16#issuecomment-1406510269

    > Anyway, have you also checked the following queries in `0_whitespace.xml` (which is now a copy of your `whitespace.xml`):
    >
    > - d4c3a72c-4458-11e6-96e2-28b2bdcff70b
    > - dbef92f4-4458-11e6-96e2-28b2bdcff70b
    > - 1f5d27dc-450c-11e6-8564-332de33a5c03
    >
    > All failing because of a missing JOIN condition. It seems they pass in your case ; I assume you have set an implicit `NATURAL JOIN`.

  - Markus - https://github.com/ivoa/lyonetia/pull/16#issuecomment-1406682153

    > Uh, ouch, you're right that these shouldn't pass, and I shouldn't
    > have dragged them from my unit tests (where they were used for
    > something completely different anyway). But unfortunately no, it's
    > not been wisdom (like default NATURAL). It's been a far too
    > permissive grammar, where I have taken a shortcut ages ago that I
    > shouldn't have taken.
    >
    > Feel free to drop these, add join conditions, or do valid="False".

  - Markus - https://github.com/ivoa/lyonetia/pull/16#issuecomment-1406705533

    > Well... Now that I consult the ADQL grammar... Ok, so I wasn't overly permissive at all, and although it looked like a shortcut at first I actually just implemented what the grammar says:
    >
    > <qualified_join> ::=
    >
    > <table_reference> [NATURAL] [<join_type>] JOIN
    >
    > <table_reference> [<join_specification>]
    >
    > So, <join_specification> happens to be optional regardless of whether or not there is a NATURAL *in our grammar*.
    >
    > Since a non-NATURAL, non-specification JOIN will later fail in the backend database, one might argue we shouldn't let it pass in ADQL either. This would mean we'd have to write something like
    >
    > <qualified_join> ::=
    >
    > <table_reference> <natural_join>
    >
    > | <table_reference> <non_natural_join>
    >
    > <natural_join> ::= NATURAL [<join_type>] JOIN <table_reference>
    >
    > <non_natural_join> ::= [<join_type>]

> JOIN <table_reference> <join_specification>
>
> (disclaimer: not tested).
>
> I'd be mildly in favour of that, enough that I'd implement it but not enough that I'd push it into the spec myself.
>
> Meanwhile, for the tests I think we ought to just put NATURAL in front of all the JOIN-s (though it's questionable whether these should be in whitespace...). That way, we don't have to invent column names and conditions on them.

- Gregory - https://github.com/ivoa/lyonetia/pull/16#issuecomment-1406870575

  > I drop these from `0_whitespace.xml` and I moved one to a new file `2_from.xml` in which I added several tests on the usage of FROM and all JOINs. It might not be exhaustive yet.
  >
  > [...]
  >
  > It seems pretty ok to me. I'll fix that in the BNF next week too, and then I'll try to see if the BNF will still be consistent.
  >
  > [...]
  >
  > I don't see what makes you hesitate. Actually, it really looks like a fix of the grammar that everyone already applied in their implementation. So, actually, this should be again, another Erratum to ADQL-2.0, isn't it?
  >
  > [...]
  >
  > You're right that when running so many queries in a real service, it is more convenient to have a well defined set of tables and columns and use NATURAL JOINs as much as possible. However, the goal of this ADQL validator is to just validate the official grammar, and not how well existing parsers or ADQL services can deal with the execution of the query. That's why, my validator is offline and does not check whether table and column names exist on a database...there is no database. So, with or without NATURAL, the validator should not complain about column and table name not existing in a database. Taking that into account, I don't think I like to alter the diversity of the test queries by forcing the usage of NATURAL JOINs everywhere. More different simple and complex queries we have, better the tests will be.

- Markus - https://github.com/ivoa/lyonetia/pull/16#issuecomment-1409894974

  > I hesitate because we're deviating from the SQL92 grammar here. It has, as ADQL 2.0 has:
  >
  > ::=
  >
  > [NATURAL] [ ] JOIN
  >
  > [ ]

> I don't have time to dig into the spec to where it says "have either NATURAL or a " -- or to ascertain that it doesn't say that and then figure out under what circumstances a single JOIN might be legal after all (perhaps as an alias for ","?).
>
> But I am saying that if we deviate from SQL92, we should be sure we know what we're doing, and I'm hesitating because this is not enough of a big deal for me to really start thinking hard. But as I said, if someone else does the deep thinking, I think I'm fine with the change per se.
>
> A 2.0 Erratum I think would be overdoing it, though. I don't think there's operational harm the way it is; bad queries just fail a bit later, that's all.
>
> ## ivoa/O1_geometrical_functions#1655f288-4d02-11e6-8c9d-819288b39233
>
> - **Description:**
>
>   > STC-S
>
> - **Query:**
>
>   ```
>   select * from foo where 1=CONTAINS(
>           REGION('Union ICRS (Position 1 2 Intersection (circle 1 2 3 box 1 2 3 4 circle 30 40 2))'),
>
>           REGION('circle GALACTIC 1 2 3')
>         )
>   ```
>
> - **Expected:** Success
> - **Status:** Currently failing in the ADQL validator because the `Union` and `Intersection` operations are not recognized by its STC-s parser.