Database - Feature #6628

port PostgreSQL native user defined functions to MariaDB

07/22/2022 11:50 AM - Eric Faulhaber

Status:	WIP	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Dănuț Filimon	**% Done:**	70%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**version:**	
vendor_id:	GCD		

Description

Related issues:

Related to Database - Feature #6348: implement support for MariaDB	**WIP**	
Related to Database - Bug #6741: Word index support for MariaDB.	**New**	**09/09/2022**

History

#1 - 07/22/2022 11:50 AM - Eric Faulhaber

- Related to Feature #6348: implement support for MariaDB added

#2 - 07/22/2022 12:11 PM - Eric Faulhaber

We need a robust library of UDFs for MariaDB to represent 4GL built-in functions used in WHERE clauses, which reference the current buffer (i.e., which need to execute at the database server).

Questions/Concerns:

- MariaDB documentation states that the languages which can be used to implement UDFs are SQL and C/C++.
  - C/C++ is not acceptable from a deployment perspective. This is not just a statement of ease of use, it simply will not be accepted for use in managed cloud deployments from a security standpoint.
  - What are the capabilities/limitations of the SQL language option? Is this strictly limited to straight SQL syntax, or is it more SQL syntax augmented with some control flow features? The MariaDB documentation I've seen so far is unclear, but the example on this page, which uses a WHILE loop construct, gives me hope: https://www.techonthenet.com/mariadb/functions.php
- We rely on many PostgreSQL built-in functions to implement the UDFs. Do we have the critical mass of built-in functions available in MariaDB needed to port the PostgreSQL UDFs?
  - These are not part of the SQL standard; a database vendor may implement as much or as little support for these as they choose.
  - Even if there is good support for many built-in functions, the implementations may be different enough to create a lot of porting effort.

#3 - 07/22/2022 12:24 PM - Igor Skornyakov

Please note also that it is makes sense to check how the words tables' support can be ported to MariaDB.

#4 - 07/22/2022 12:40 PM - Greg Shah

Looping in SQL can be simulated using Common Table Expressions which is a SQL standard and exists in MariaDB, PostgresSQL, SQLServer, H2 (experimental). I have no idea how performant it is or if it meets our needs from a control flow perspective.

I suspect that error handling will be a problem.

#5 - 07/22/2022 12:42 PM - Eric Faulhaber

Igor Skornyakov wrote:

> Please note also that it is makes sense to check how the words tables' support can be ported to MariaDB.

Yes. While there is no immediate requirement for word index support (the initial target project does not use them), we will need to implement this ultimately, to consider the support for MariaDB complete.

What are the technical prerequisites a database implementation must have for word index support?

#6 - 07/22/2022 12:55 PM - Eric Faulhaber

Igor, can you think of a way to generate a list of the PostgreSQL built-in functions (including signatures) upon which we rely in the current implementation of the native UDFs? Or is this just a matter of manually reviewing the UDF code?

#7 - 07/22/2022 01:01 PM - Eric Faulhaber

Greg Shah wrote:

> I suspect that error handling will be a problem.

I guess you're right, based on the answers here: https://stackoverflow.com/questions/465727/how-to-raise-an-error-within-a-mysql-function. The one that starts, "Why not just store a VARCHAR in a declared INTEGER variable?" is a clever (if hacky) workaround for this very fundamental limitation. Might be we have to do something similar, and clean up the reporting on the FWD server side.

#8 - 07/22/2022 01:01 PM - Igor Skornyakov

Eric Faulhaber wrote:

> What are the technical prerequisites a database implementation must have for word index support?

As far as I understand, not that much is required from the database. What comes to my mind now is:

- Common Tables Expression (WITH clause) support. We use pretty standard form of this
- Subquery (query result) instead of the table name in the FROM clause. I'm not sure how the current way we use it complies to SQL standard (and supported by MariaDB).
- ON INSERT/ON UPDATE trigger
- Efficient words function returning the table (set of records).

Actually the templates of the SQL queries used for re-writing CONTAINS are described in details (with samples) in the corresponding Wiki section - https://proj.goldencode.com/projects/p2j/wiki/Word_Index_and_CONTAINS_Operator

#9 - 07/22/2022 01:05 PM - Eric Faulhaber

Eric Faulhaber wrote:

> Greg Shah wrote:
>
> > I suspect that error handling will be a problem.
>
> I guess you're right, based on the answers here: https://stackoverflow.com/questions/465727/how-to-raise-an-error-within-a-mysql-function. The one that starts, "Why not just store a VARCHAR in a declared INTEGER variable?" is a clever (if hacky) workaround for this very fundamental limitation. Might be we have to do something similar, and clean up the reporting on the FWD server side.

A better option seems to be the SIGNAL statement (https://mariadb.com/kb/en/signal/), which the documentation states "can be used anywhere", but I don't know if that includes inside a function body. Need to test this.

#10 - 07/22/2022 01:06 PM - Igor Skornyakov

Eric Faulhaber wrote:

> Igor, can you think of a way to generate a list of the PostgreSQL built-in functions (including signatures) upon which we rely in the current implementation of the native UDFs? Or is this just a matter of manually reviewing the UDF code?

OK, I will think in a background how to do it.
In fact I expect that for ~90% of UDFs re-writing to another procedurial SQL dialect will be more or less straightforward. However for the rest 10% is can be a challenge.

#11 - 07/22/2022 02:20 PM - Eric Faulhaber

So far, so good...

```
MariaDB [test]> delimiter $$
MariaDB [test]> create or replace function raiseError ( errno int, message varchar(255) ) returns int
    -> begin
    ->    signal sqlstate '55000'
    ->    set mysql_errno = errno, message_text = message;
    ->    return 0;
    -> end;
    -> $$
```

```
Query OK, 0 rows affected (0.007 sec)

MariaDB [test]> delimiter ;
MariaDB [test]> select raiseError(34, "Blurp");
ERROR 34 (55000): Blurp
```

#12 - 07/22/2022 02:25 PM - Igor Skornyakov

Eric Faulhaber wrote:

> So far, so good...
>
> [...]

Please note the PostreSQL UDF raise both ERROR and NOTICE. The later does not result in SQL error but populates SQLWarings that should cause application warnings (not yet implemented).

#13 - 07/23/2022 04:28 AM - Igor Skornyakov

Eric Faulhaber wrote:

> Igor, can you think of a way to generate a list of the PostgreSQL built-in functions (including signatures) upon which we rely in the current implementation of the native UDFs? Or is this just a matter of manually reviewing the UDF code?

Please find a complete (I hope) list of PostgreSQL built-in functions used in FWD UDFs below:

- abs(numeric)
- age(timestamp, timestamp)
- array_agg(expression)
- array_length(anyarray, int)
- array_position(anyarray, anyelement [, int])
- array_to_string(anyarray, text [, text])
- chr(int)
- coalesce(value [, ...])
- convert_from(string bytea, src_encoding name)
- convert_to(string text, dest_encoding name)
- date_part(text, interval)
- exists(subquery)
- extract(field FROM source)
- greatest(value [, ...])
- left(str text, n int)
- length(string)
- log(numeric)
- lower(string)
- lpad(string text, length int [, fill text])
- ltrim(string text [, characters text])
- make_date(year int, month int, day int)
- make_timestamp(year int, month int, day int, hour int, min int, sec double precision)
- max(expression)
- now()
- nullif(value1, value2)
- octet_length(string)
- position(substring in string)
- regexp_matches(string text, pattern text [, flags text])

- regexp_replace(string text, pattern text, replacement text [, flags text])
- regexp_split_to_array(
- repeat(string text, number int)
- replace(string text, from text, to text)
- right(str text, n int)
- round(numeric)
- row_number()
- rpad(string text, length int [, fill text])
- rtrim(string text [, characters text])
- scale(numeric)
- string_agg(expression, delimiter)
- string_to_array(text, text [, text])
- strpos(string, substring)
- substring(string, from [, count])
- to_char(date, text)
- to_char(numeric, text)
- translate(string text, from text, to text)
- trim([leading | trailing | both] [characters] from string)
- trunc(numeric)
- unnest(anyarray)
- upper(string)

#14 - 07/24/2022 05:08 AM - Igor Skornyakov

It looks that in migration of the PostgreSQL native UDFs to MariaDB the most efforts will require those UDFs that use arrays (I understand that MariaDB does not have arrays),
It is matcheslist and all format-driven conversions. Some changes may also be needed for the code that uses regular expressions.
Tha rest should not be a big deal.

#15 - 07/29/2022 01:25 AM - Eric Faulhaber

- Assignee set to Igor Skornyakov

Igor Skornyakov wrote:

> I understand that MariaDB does not have arrays

Not as a first class data type, no, but please see #6418. Although we are having to implement extent fields as described there, I don't know that this will be helpful at the UDF implementation level.

#16 - 07/31/2022 03:41 PM - Igor Skornyakov

- Status changed from New to WIP

#17 - 08/01/2022 04:14 AM - Igor Skornyakov

Which version of MariaDB should I work with?
Thank you.

#18 - 08/01/2022 12:45 PM - Eric Faulhaber

Igor Skornyakov wrote:

> Which version of MariaDB should I work with?

The most recent production build (10.8, AFAIK). Ovidiu, which are you using?

#19 - 08/01/2022 01:27 PM - Igor Skornyakov

Working with version 10.8.3

#20 - 08/01/2022 02:22 PM - Ovidiu Maxiniuc

I used the following command:

```
$ sudo apt install mariadb-server
```

That installed for me:

```
Server version: 10.3.34-MariaDB-0ubuntu0.20.04.1
```

I do not know why is not the very last, I think that was the standard for the 20.04' repositories. In 22.04 LTE, MariaDb 10.8.3 might be the default. I will investigate later whether the newer versions brings something we can use.

#21 - 08/01/2022 02:32 PM - Igor Skornyakov

Ovidiu Maxiniuc wrote:

> I used the following command:
> [...]

> That installed for me:
> [...]
>
> I do not know why is not the very last, I think that was the standard for the 20.04' repositories. In 22.04 LTE, MariaDb 10.8.3 might be the default.
> I will investigate later whether the newer versions brings something we can use.

On main machine this method failed. And I've seen somewhere that standard Ubuntu repo contains version 10.3.
So I've used the procedure described here: https://www.linuxcapable.com/install-upgrade-mariadb-10-8-on-ubuntu-20-04-lts/

#22 - 08/03/2022 12:11 PM - Igor Skornyakov

Have we already made a decision about datatypes' mapping for MariaDB?
At this moment the most important for me are string data types.
Thank you.

#23 - 08/03/2022 12:52 PM - Eric Faulhaber

Please see #6348-7. In fact, it would be good to review that whole task.

#24 - 08/03/2022 01:10 PM - Igor Skornyakov

Eric Faulhaber wrote:

> Please see #6348-7. In fact, it would be good to review that whole task.

Thank you, reviewing.

#25 - 08/05/2022 12:04 PM - Igor Skornyakov

There is a about a couple of dozens of PostgreSQL native UDFs (~10%) that heavily use arrays and/or user-defined data types (UDT).

I've not found a good "universal" replacement for these PostgreSQL constructs in MariaDB except JSON datatype.

I completely understand the concerns described in #6418-5 but I think that it is an acceptable approach at the first step.

After all UDFs will be implemented and pass the compatibility test suite we can apply UDF-specific technique for creating more efficient logic if the performance tests will reveal significant performance problems.

Unfortunately those UDFs which rely on PostgreSQL regexp functions should be re-implement more or less from scratch sine regext support in MariaDB is very basic comparing to one in PostgreSQL.

What do you think about using JSON datatype as described above?
Thank you.

#26 - 08/08/2022 11:55 PM - Eric Faulhaber

If the JSON array approach seems to be the only viable option, I think we have to use it for now. Another alternative is to implement some of the UDFs in C/C++, but we really want to avoid this. It is problematic in other ways, not the least of which is that it limits our deployment options.

#27 - 08/10/2022 08:01 AM - Igor Skornyakov

At this moment I see the following plan

1. Create an AWK script for automatic conversion of the udfs.sql script for PostgreSQL to the MariaDB version. The target script should contain valid MariaDB function declarations with correct arguments and return type and empty body. The body of the PostgreSQL version will be added as comment to assist in manual conversion.
2. Start conversion PostgreSQL UDFs to MariaDB starting with 'simple' ones. I expect that it will be possible to convert ~ 10-20 UDFs per day.
3. Convert remaining 'complicated' UDFs related format-driven conversions and word tables' support.
4. Modify my compatibility test suite to be able to run compatibility tests at least once a week to test the newly converted UDFs. Of course this will be possible when conversion/runtime/import support for MariaDB will be ready. I've invented a simple amendment of the tests' generation to support such incremental testing.

#28 - 08/10/2022 01:24 PM - Igor Skornyakov

I've realized that in MariaDB we cannot have two UDFs with the same name but different signatures.
This requires changes in the UDFs naming conversion. This affects the query conversion/rewriting.
Do we already have a solution?
Thank you.

#29 - 08/10/2022 01:38 PM - Igor Skornyakov

Igor Skornyakov wrote:

> I've realized that in MariaDB we cannot have two UDFs with the same name but different signatures.
> This requires changes in the UDFs naming conversion. This affects the query conversion/rewriting.
> Do we already have a solution?
> Thank you.

I suggest to append and underscore and an encoded arguments' types to the MariaDB UDF name to make them unique.
Below are all types of PostgreSQL UDFs arguments and suggested one-letter code for it:

bigint	L
boolean	B
date	D
integer	I
numeric	N
text	T
text[]	A
timestamp without time zone	S
time without time zone	M
udf.time_format	F

For example the name of the MariaDB counterpart for udf.tostring(v boolean, fmt text) will be tostring_BT.
Is it OK?
Thank you.

#30 - 08/10/2022 02:06 PM - Eric Faulhaber

We had a similar issue with H2 and we resolved it by prepending an underscore followed by a unique integer (for that UDF). Not the most elegant naming convention, but it was hidden from plain view by virtue of the fact that H2 generally is embedded in the JVM and the schema is not used externally.

The UDF names in the converted where clauses in the application source remain the same. The overloading is handled in the HQLPreprocessor (we really need to change that name), by mapping the overloaded name to the specific UDF name for a specific signature.

I'm ok with your naming convention, though where the mapped data types differ in name for Maria, please adjust the letter codes accordingly, so they make sense to the degree possible. Is text the only type which can be passed as an array?

Please leverage the infrastructure that already exists in the runtime for the overloading. I don't think it is hard-coded to H2; I think we used the dialect architecture. If you do find some assumptions that are specific to the H2 mapping, please generalize them using the dialect idiom.

#31 - 08/10/2022 02:39 PM - Igor Skornyakov

Eric Faulhaber wrote:

> We had a similar issue with H2 and we resolved it by prepending an underscore followed by a unique integer (for that UDF). Not the most elegant naming convention, but it was hidden from plain view by virtue of the fact that H2 generally is embedded in the JVM and the schema is not used externally.

Yes, I've seen these numeric suffixes, but to be honest have never completely understood how it works exactly.

> The UDF names in the converted where clauses in the application source remain the same. The overloading is handled in the HQLPreprocessor (we really need to change that name), by mapping the overloaded name to the specific UDF name for a specific signature.
>
> I'm ok with your naming convention, though where the mapped data types differ in name for Maria, please adjust the letter codes accordingly, so they make sense to the degree possible. Is text the only type which can be passed as an array?

In fact the MariaDB data types a not very different from PostgreSQL ones. See the complete table:

PostgresSQL	MariaDB	Code
bigint	BIGINT	L
boolean	BOOLEAN	B
date	DATE	D
integer	INTEGER	I
numeric	DECIMAL (65, 38)	N
text	TEXT	T
text[]	JSON	A
timestamp without time zone	TIMESTAMP	S
time without time zone	time	M
udf.time_format	JSON	F

The only array type used in the UDFs is text[]

> Please leverage the infrastructure that already exists in the runtime for the overloading. I don't think it is hard-coded to H2; I think we used the dialect architecture. If you do find some assumptions that are specific to the H2 mapping, please generalize them using the dialect idiom.

OK. But at this moment I'm working on UDFs themselves.

#32 - 08/10/2022 03:12 PM - Igor Skornyakov

Created MariaDB UDFs templates.

Committed to 3821c/14145.

#33 - 08/10/2022 05:24 PM - Ovidiu Maxiniuc

Igor Skornyakov wrote:

> I'm ok with your naming convention, though where the mapped data types differ in name for Maria, please adjust the letter codes accordingly, so they make sense to the degree possible. Is text the only type which can be passed as an array?

In fact the MariaDB data types a not very different from PostgreSQL ones. See the complete table:

PostgresSQL	MariaDB	Code
bigint	BIGINT	L
boolean	BOOLEAN	B
date	DATE	D
integer	INTEGER	I
numeric	DOUBLE	N
text	TEXT	T
text[]	JSON	A
timestamp without time zone	TIMESTAMP	S
time without time zone	time	M
udf.time_format	JSON	F

The only array type used in the UDFs is text[]

I used a very similar (if not identically approach) for SQL Server which as the same constraints. For details please see SQLServerHelper.getSqlDecoration() method. The letters are almost the same, but in lowercase. Only the types used in the existing UDFs at that time are present, but the list can be updated, of course.

#34 - 08/11/2022 01:14 AM - Eric Faulhaber

Igor Skornyakov wrote:

> [...]
> In fact the MariaDB data types a not very different from PostgreSQL ones. See the complete table:
>
> | PostgresSQL | MariaDB | Code |
> |---|---|---|
> | bigint | BIGINT | L |
> | boolean | BOOLEAN | B |
> | date | DATE | D |
> | integer | INTEGER | I |
> | numeric | DOUBLE | N |
> | text | TEXT | T |
> | text[] | JSON | A |
> | timestamp without time zone | TIMESTAMP | S |
> | time without time zone | time | M |
> | udf.time_format | JSON | F |
>
> [...]

I just looked at this again and noticed that some of the mappings aren't the same as those specified in #6348-7. We should not be using double; this should be decimal. Does varchar(#) automatically widen to text when passed as a function parameter?

#35 - 08/11/2022 03:13 AM - Igor Skornyakov

Eric Faulhaber wrote:

> Igor Skornyakov wrote:
> I just looked at this again and noticed that some of the mappings aren't the same as those specified in #6348-7. We should not be using double; this should be decimal. Does varchar(#) automatically widen to text when passed as a function parameter?

I've changed DOUBLE TO NUMERIC.
I've noticed no problems with calling functions with text parameter supplying varchar(n) value so far.

#36 - 08/11/2022 12:45 PM - Igor Skornyakov

It appears that we should use DECIMAL(65, 38) instead of NUMERIC in UDFs.

#37 - 08/11/2022 01:59 PM - Igor Skornyakov

In 3821c/14150. **55** of **219** UDFs are ported from PostrgreSQL to MariaDB.

Please do not be too optimistic. The ported UDFs are very simple ones.

#38 - 08/11/2022 02:05 PM - Eric Faulhaber

Igor Skornyakov wrote:

> It appears that we should use DECIMAL(65, 38) instead of NUMERIC in UDFs.

Shouldn't we be using DECIMAL(50, 10) to match the 4GL's limits? To the degree the original built-in functions implemented in Java dealt with 4GL decimal values, we would have applied these limits. These UDF implementations should work the same way.

#39 - 08/11/2022 02:13 PM - Igor Skornyakov

Eric Faulhaber wrote:

> Igor Skornyakov wrote:
>
>> It appears that we should use DECIMAL(65, 38) instead of NUMERIC in UDFs.
>
> Shouldn't we be using DECIMAL(50, 10) to match the 4GL's limits? To the degree the original built-in functions implemented in Java dealt with 4GL decimal values, we would have applied these limits. These UDF implementations should work the same way.

I'm not sure. It seems that MariaDB is not very strict about this. But change from (65, 38) to (50, 10) is a one-step operation.

#40 - 08/11/2022 02:17 PM - Eric Faulhaber

If I understand your process correctly, it looks like you are doing an automated port of the "scaffolding" from the PostgreSQL implementations, generating the CREATE FUNCTION statements with remapped syntax and data types, but with the function bodies commented out. Then, you review a particular function and manually reimplement the commented body using MariaDB constructs/facilities. Is that roughly correct?

#41 - 08/11/2022 02:19 PM - Igor Skornyakov

Eric Faulhaber wrote:

> If I understand your process correctly, it looks like you are doing an automated port of the "scaffolding" from the PostgreSQL implementations, generating the CREATE FUNCTION statements with remapped syntax and data types, but with the function bodies commented out. Then, you review a particular function and manually reimplement the commented body using MariaDB constructs/facilities. Is that roughly correct?

Exactly. Any objections?

#42 - 08/11/2022 02:27 PM - Eric Faulhaber

Igor Skornyakov wrote:

> Exactly. Any objections?

No, seems like a good approach.

I would like to get the infrastructure in place to install the UDFs (even in this early stage) from FWD, to allow testing as soon as possible. Is there anything that needs to be done to integrate this into the work done for #6574?

#43 - 08/11/2022 02:32 PM - Igor Skornyakov

Eric Faulhaber wrote:

> Igor Skornyakov wrote:
>
> > Exactly. Any objections?
>
> No, seems like a good approach.
>
> I would like to get the infrastructure in place to install the UDFs (even in this early stage) from FWD, to allow testing as soon as possible. Is there anything that needs to be done to integrate this into the work done for #6574?

Well, apart from runtime support for MariaDB UDFs (such as UDFs names support), we need at least a MariaDB version of the ScriptRunner. I put words tables' support aside.
And, of course, we need MariaDB DDL generation.

#44 - 08/12/2022 03:19 PM - Igor Skornyakov

In 3821c/14153. **157** of **219** UDFs are ported from PostrgreSQL to MariaDB.

So, all 'simple' UDFs have been ported. The remaining ones will take substantially more efforts.

I've also replaced DECIMAL(65, 38) with DECIMAL(50, 10)

#45 - 08/14/2022 11:40 AM - Igor Skornyakov

In 3821c/14154. **174** of **219** UDFs are ported from PostrgreSQL to MariaDB.

Started working on format-driven conversions.

#46 - 08/14/2022 11:55 AM - Igor Skornyakov

According to the MariaDB documentation its date functions do not work correctly with non-gregorian dates (before October 15, 1582).
So at this moment FWD todate(integer) and toint(date) UDFs work only with gregorian dates.
Is it OK?
Thank you.

#47 - 08/15/2022 01:03 PM - Eric Faulhaber

Igor Skornyakov wrote:

> According to the MariaDB documentation its date functions do not work correctly with non-gregorian dates (before October 15, 1582).
> So at this moment FWD todate(integer) and toint(date) UDFs work only with gregorian dates.
> Is it OK?
> Thank you.

The parameter passed to todate(integer) is the number of days since 12/31/-4714 (the Julian day number), and toint(date) returns the Julian day number, given a date parameter. How would this conversion work internally?

#48 - 08/15/2022 01:10 PM - Igor Skornyakov

Eric Faulhaber wrote:

> The parameter passed to todate(integer) is the number of days since 12/31/-4714 (the Julian day number), and toint(date) returns the Julian day number, given a date parameter. How would this conversion work internally?

At this moment these UDFs are implemented like this:

```
CREATE OR REPLACE FUNCTION todate_I(d INTEGER)
RETURNS DATE
DETERMINISTIC
BEGIN
```

```
   declare gd integer;
   set gd = d - 2299162;
   if gd < 0 then
      SIGNAL SQLSTATE 'FWD00' SET MESSAGE_TEXT =  'Dates before October 15, 1582 are not yet supported';
   end if;
   RETURN cast('1582-10-15' as date) + interval gd day;
END;
$$

CREATE OR REPLACE FUNCTION toint_D(dt DATE)
RETURNS BIGINT
DETERMINISTIC
BEGIN
   if dt < cast('1582-10-15' as date) then
      SIGNAL SQLSTATE 'FWD00' SET MESSAGE_TEXT =  'Dates before October 15, 1582 are not yet supported';
   end if;
   RETURN 2299162 + datediff(dt, cast('1582-10-15' as date) );
END;
$$
```

#49 - 08/15/2022 01:20 PM - Eric Faulhaber

I initially was confused by the statement:

> So at this moment FWD todate(integer) and toint(date) UDFs work only with gregorian dates.

But if I understand now, the functions work with Julian day values, just not those that would correspond with dates before October 15, 1582 in the Gregorian calendar, correct? I am ok with that limitation, as long as it is documented.

#50 - 08/15/2022 01:28 PM - Igor Skornyakov

Eric Faulhaber wrote:

> But if I understand now, the functions work with Julian day values, just not those that would correspond with dates before October 15, 1582 in the Gregorian calendar, correct? I am ok with that limitation, as long as it is documented.

Yes, this is correct. In fact the algorithm for all julian day/julian date values is not very complicated. I can implement it later.

#51 - 08/15/2022 07:21 PM - Ovidiu Maxiniuc

Igor,
Please do a check-out of the hotel_gui project (~/secure/code/p2j_repo/samples/hotel_gui/). Use the latest FWD (r14161). You will have to enable database setup for MariaDb by editing the build.properties and update the following lines:

```
db.postgresql=false
db.mariadb=true
db.port=3306
```

Of course, the credentials will possibly need adjustments.

Do a full build and deploy of the project (ant deploy.all).

When I am running ant create.db.mariadb I get the following messages:

```
sql.udfs.mariadb:
     [java] Exception in thread "main" java.lang.IllegalStateException: Cannot open script [/udf/mariadb/udfs.
sql]
     [java]     at com.goldencode.p2j.persist.deploy.ScriptRunner.apply(ScriptRunner.java:296)
     [java]     at com.goldencode.p2j.persist.deploy.ScriptRunner.main(ScriptRunner.java:197)

BUILD FAILED
/home/om/workspaces/hotel-maria/build.xml:893: The following error occurred while executing this line:
/home/om/workspaces/hotel-maria/build.xml:828: The following error occurred while executing this line:
/home/om/workspaces/hotel-maria/build.xml:830: The following error occurred while executing this line:
/home/om/workspaces/hotel-maria/build_db.xml:265: The following error occurred while executing this line:
/home/om/workspaces/hotel-maria/build_db.xml:277: Java returned: 1
```

Could you take a look over ScriptRunner and make it work with MariaDB dialect? Without this, we are unable to install the UDFs at all :(.

Note: it seems strange that the runner wants to read the file from that absolute path. Maybe am am doing something wrong. Please let me know if you have any idea in this regard.
Thank you!

#52 - 08/15/2022 11:17 PM - Eric Faulhaber

The following is a list of 4GL builtin functions used within WHERE clauses, based on their presence in the application which first needs to run with MariaDB as the backend:

1. input()
2. recid()
3. entry()
4. rowid()
5. lookup()
6. string()
7. substring()
8. today()
9. index()
10. num-entries()
11. decimal()
12. int64()
13. trim()
14. r-index()
15. integer()
16. frame-line()
17. frame-down()
18. maximum()
19. truncate()
20. can-do()
21. chr()
22. replace()
23. valid-handle()
24. program-name()
25. ldbname()
26. caps()
27. weekday()
28. length()
29. right-trim()
30. substitute()
31. minimum()
32. available()
33. sdbname()
34. interval()
35. now()
36. userid()
37. lc()
38. pdbname()

This list was taken from a FWD Analytics report, and the order of the list represents the frequency of use of these builtin functions (most frequently used functions first). As such, it roughly represents the priority of the corresponding implementation of the UDFs for MariaDB.

However, the report only recognizes the *presence* of a builtin function in a WHERE, clause. It *does not* differentiate between a builtin function which will be executed in the FWD server as part of a query substitution parameter (i.e., once per query, before the query is executed), and one which must execute as a database server-side UDF (i.e., once per row, during query execution). The report was too voluminous, such that tracking down every use of each of these functions to filter out the ones that were only used in query substitution parameter sub-expressions was not practical. However, if you have questions about specific functions (e.g., if the implementation of some UDF seem particularly difficult and you want to know if a particular subset of functions is actually used as UDFs), please ask.

#53 - 08/15/2022 11:22 PM - Eric Faulhaber

Ovidiu Maxiniuc wrote:

> Igor,
> Please do a check-out of the hotel_gui project (~/secure/code/p2j_repo/samples/hotel_gui/). Use the latest FWD (r14161). You will have to enable database setup for MariaDb by editing the build.properties and update the following lines:
> [...]Of course, the credentials will possibly need adjustments.
>
> Do a full build and deploy of the project (ant deploy.all).
>
> When I am running ant create.db.mariadb I get the following messages:
> [...]
>
> Could you take a look over ScriptRunner and make it work with MariaDB dialect? Without this, we are unable to install the UDFs at all :(.

Igor, getting this going is high priority, as we need to get as much early testing going as possible before Ovidiu leaves for holiday next week. Please address this before continuing with the implementation of individual UDFs.

> Note: it seems strange that the runner wants to read the file from that absolute path.

This does seem strange, as the path should be relative to the p2j root path.

#54 - 08/16/2022 01:31 AM - Igor Skornyakov

OK. Switching to the ScriptRunner and most commonly used UDFs.
Please note however that we also need UDFs names re-writing. In particular, when looking at the error in #6348-28 I see that it is not done yet.

#55 - 08/16/2022 02:41 AM - Igor Skornyakov

Changed build.gradle to add MariaDB udfs.sql to the p2j.jar in 3821c/14165.

#56 - 08/16/2022 10:42 AM - Ovidiu Maxiniuc

Igor Skornyakov wrote:

> OK. Switching to the ScriptRunner and most commonly used UDFs.
> Please note however that we also need UDFs names re-writing. In particular, when looking at the error in #6348-28 I see that it is not done yet.

I can do that while you handle the ScriptRunner. Let me know if you already started this so we do not overlap.

#57 - 08/16/2022 12:17 PM - Igor Skornyakov

Ovidiu Maxiniuc wrote:

> Igor Skornyakov wrote:
>
> > OK. Switching to the ScriptRunner and most commonly used UDFs.
> > Please note however that we also need UDFs names re-writing. In particular, when looking at the error in #6348-28 I see that it is not done yet.
>
> I can do that while you handle the ScriptRunner. Let me know if you already started this so we do not overlap.

No, I do not work on this. In fact I've planned to finish with porting UDFs first.
Thank you.

#58 - 08/16/2022 01:10 PM - Eric Faulhaber

Igor Skornyakov wrote:

> No, I do not work on this. In fact I've planned to finish with porting UDFs first.

Ovidiu can handle implementing the overloading logic, but please make sure he has whatever else he may need to integrate the UDFs with his overall MariaDB support (including any ScriptRunner changes). This is top priority, as we need him to be able to move forward testing Hotel GUI. Then return to porting the UDFs themselves. Thank you.

#59 - 08/16/2022 01:20 PM - Igor Skornyakov

Refactored ScriptRunner moving dialect-specific logic into the corresponding subclasses. Added support for MariaDB (including UDFs creation on the server startup if those ones are not found)
Committed to 3821c/14167.

Tested with Hotel app.

#60 - 08/16/2022 07:39 PM - Ovidiu Maxiniuc

Igor,

I implemented the decoration of the UDFs as described in note-31. I encountered some differences. I know you are working on it, I am putting the diff here just to have a current status: UDF name diffUDF name diff

```
--- <functions registered into DB (select SPECIFIC_NAME from information_schema.routines)>
+++ <functions required by FQL>
@@ -7,10 +7,19 @@
 begins_TT
 begins_TTB
 caps_T
+checkerror_BB
 concat_TT
+contains_TT
+contains_TTB
 datespan_DD
 divide_II
+divide_IL
+divide_IN
+divide_LI
 divide_LL
+divide_LN
+divide_NI
+divide_NL
 divide_NN
 entryin_IT
 entryin_ITT
@@ -19,24 +28,31 @@
 eq_BB
 eq_DD
 eq_II
+eq_IL
+eq_IN
+eq_LI
 eq_LL
+eq_LN
+eq_NI
+eq_NL
 eq_NN
 eq_SS
 eq_TT
 fill_TI
 fill_TL
+fwdsession_
 getday_D
 getday_S
-getFWDVersion
+getfwdversion_
 getinterval_DDT
 getinterval_DST
 getinterval_SDT
 getinterval_SST
 getmonth_D
 getmonth_S
-getmtime
+getmtime_
 getmtime_S
-gettimezone
+gettimezone_
 getweekday_D
 getweekday_S
 getyear_D
@@ -44,31 +60,52 @@
 gte_BB
 gte_DD
 gte_II
+gte_IL
+gte_IN
+gte_LI
 gte_LL
+gte_LN
+gte_NI
+gte_NL
```

```
 gte_NN
 gte_SS
 gte_TT
 gt_BB
 gt_DD
 gt_II
+gt_IL
+gt_IN
+gt_LI
 gt_LL
+gt_LN
+gt_NI
+gt_NL
 gt_NN
 gt_SS
 gt_TT
 indexof_TT
-indexof_TTB
+indexof_TTI
+indexof_TTIB
+indexof_TTL
+indexof_TTLB
 indexof_TTN
 indexof_TTNB
+initerror_B
 lengthof_T
 lengthof_TT
 lookup_TT
-lookup_TTB
 lookup_TTT
 lookup_TTTB
 lte_BB
 lte_DD
 lte_II
+lte_IL
+lte_IN
+lte_LI
 lte_LL
+lte_LN
+lte_NI
+lte_NL
 lte_NN
 lte_SS
 lte_TT
@@ -77,55 +114,111 @@
 lt_BB
 lt_DD
 lt_II
+lt_IL
+lt_IN
+lt_LI
 lt_LL
+lt_LN
+lt_NI
+lt_NL
 lt_NN
 lt_SS
 lt_TT
 matcheslist_TT
-matcheslist__TT
 matches_TT
 matches_TTB
 matches_TTBB
-matches__TTBB
 maximum_DD
+maximum_II
+maximum_IL
+maximum_IN
+maximum_LI
+maximum_LL
+maximum_LN
+maximum_NI
+maximum_NL
 maximum_NN
 maximum_SS
 maximum_TT
 minimum_DD
+minimum_II
+minimum_IL
+minimum_IN
+minimum_LI
+minimum_LL
+minimum_LN
+minimum_NI
+minimum_NL
 minimum_NN
 minimum_SS
 minimum_TT
+minusmillis_SI
+minusmillis_SL
 minusmillis_SN
+minus_DI
+minus_DL
 minus_DN
 minus_II
+minus_IL
+minus_IN
+minus_LI
 minus_LL
+minus_LN
+minus_NI
+minus_NL
 minus_NN
 modulo_II
+modulo_IL
+modulo_IN
+modulo_LI
 modulo_LL
+modulo_LN
+modulo_NI
+modulo_NL
 modulo_NN
 multiply_II
+multiply_IL
+multiply_IN
+multiply_LI
 multiply_LL
+multiply_LN
+multiply_NI
+multiply_NL
 multiply_NN
 ne_BB
 ne_DD
 ne_II
+ne_IL
+ne_IN
+ne_LI
 ne_LL
+ne_LN
+ne_NI
+ne_NL
 ne_NN
 ne_SS
 ne_TT
 numentries_T
 numentries_TT
-parse_time_format_T
 plusmillis_NS
+plusmillis_SI
+plusmillis_SL
 plusmillis_SN
+plus_DI
+plus_DL
 plus_DN
+plus_ID
 plus_II
+plus_IL
+plus_IN
+plus_LD
+plus_LI
```

```
 plus_LL
+plus_LN
 plus_ND
+plus_NI
+plus_NL
 plus_NN
-regex1_TB
-regex_T
-regex_TB
 replacetext_TTT
 reportprecisionscale_N
 rounddec_N
@@ -142,8 +235,6 @@
 substringof_TLL
 substringof_TLLT
 timespan_SS
-timetostring_MTI
-time_format_regexp_
 tochr_I
 tochr_L
 todatetime_D
@@ -202,19 +293,15 @@
 tostring_IT
 tostring_L
 tostring_LT
-tostring_LTI
-tostring_MT
-tostring_MTI
-tostring_MTT
+tostring_LTT
 tostring_N
 tostring_NT
+tostring_S
 tostring_ST
 tostring_STT
 tostring_STTI
 tostring_T
 tostring_TT
-tostring__TT
-totimestring_LTI
 trimws_T
 trimws_TT
 words_T
```

#61 - 08/17/2022 01:30 AM - Igor Skornyakov

Ovidiu Maxiniuc wrote:

> Igor,
>
> I implemented the decoration of the UDFs as described in note-31. I encountered some differences. I know you are working on it, I am putting the diff here just to have a current status: {{collapse(UDF name diff)
> [...]
> }}

Thank you, Ovidiu.
I will take a look. Please note however, that I've generated templates for MariaDB UDFs from PostgreSQL ones. This is why we have no e.g. eq_LI UDF. only eq_LL and eq_II.

#62 - 08/17/2022 02:48 AM - Igor Skornyakov

Added placeholder for the error handling UDFs in 3821c/14169.

Please note that some native UDFs (such as totimestring) are not used directly by FWD code but are called from other UDFs.
contains UDFs are used only with H2. For PostgreSQL and MariaDB word tables are used to support CONTAINS operator.
I will add additional version of the missing UDFs.

Ovidiu,
How can I create a diff file from #6628-60 to check that I have not missed something?
Thank you.

#63 - 08/17/2022 03:40 AM - Igor Skornyakov

Please note also that I decided do not add underscore to the names of UDFs w/o argument. So it should be getfwdversion, not getfwdversion_.
Is it OK?
Thank you.

#64 - 08/17/2022 06:54 AM - Igor Skornyakov

Added additional UDFs' signatures in 3821c/14171.

fwdsession UDF does not exist for PostgreSQL. Actually I do not see how it can be implemented in a native mode.
See also #6628-61 - #6628-63

#65 - 08/17/2022 08:19 AM - Greg Shah

If I understand correctly, different database "dialects" (in FWD) will have different function names for the same 4GL built-in function. Is that right?

- PostgreSQL uses overloading and schema names to qualify functions, with differences between Java and native SQL functions
- H2 uses Java functions, I'm not sure how the naming works
- SQLServer has no overloading so there is some signature string appended
- MariaDB has no overloading so there is some signature string appended

Consider that we will create problems for direct SQL usage in the future if we have different function names. This will affect future developers maintaining post-conversion and/or working on new code that uses the same database back end (possibly manually migrating logic that comes from the converted application). We really should have a single set of names that can be used across all databases and both Java and native functions. I know that we try to take advantage of overloading/schemas in PostgreSQL but that has a real cost long term. It seems now is a good time to clean this up. It would also have the benefit of avoiding a more complex mapping of these names at runtime. At least it would be very slightly less work to process queries.

This assumes that a single set of names can be formulated to work across all databases, which seems very possible. If it is already working this way, then I apologize for my misunderstanding.

#66 - 08/17/2022 09:12 AM - Igor Skornyakov

Greg Shah wrote:

> If I understand correctly, different database "dialects" (in FWD) will have different function names for the same 4GL built-in function. Is that right?
>
> - PostgreSQL uses overloading and schema names to qualify functions, with differences between Java and native SQL functions
> - H2 uses Java functions, I'm not sure how the naming works
> - SQLServer has no overloading so there is some signature string appended
> - MariaDB has no overloading so there is some signature string appended
>
> Consider that we will create problems for direct SQL usage in the future if we have different function names. This will affect future developers maintaining post-conversion and/or working on new code that uses the same database back end (possibly manually migrating logic that comes from the converted application). We really should have a single set of names that can be used across all databases and both Java and native functions. I know that we try to take advantage of overloading/schemas in PostgreSQL but that has a real cost long term. It seems now is a good time to clean this up. It would also have the benefit of avoiding a more complex mapping of these names at runtime. At least it would be very slightly less work to process queries.
>
> This assumes that a single set of names can be formulated to work across all databases, which seems very possible. If it is already working this way, then I apologize for my misunderstanding.

This is a good point. However, I suggest to finish with MariaDB first. The backport to PostgreSQL should be simple.

#67 - 08/17/2022 11:05 AM - Eric Faulhaber

Devil's advocate point: there already are significant differences in syntax and builtin function names across database vendors, which any hand-written SQL would need to take into account.

The UDFs we are porting here represent legacy, 4GL builtin functions. They exist only to support references to those existing, legacy functions in converted WHERE clauses, but one would not choose to use these in hand-written SQL. Applications which have some existing dependency on hand-written SQL external to the 4GL code already have had to do without them until now.

Appending some signature code to a base function name to represent overloaded signatures for these legacy functions is a somewhat ugly compromise we use to work around databases which don't support function overloading naturally. I've been trying to avoid exposing this through the converted application, because I think seeing todate(...) in a converted WHERE clause, for example, is a lot more appealing and less confusing than seeing todate_LII(...).

That being said, I agree that the naming across databases which do not support overloading should be consistent.

#68 - 08/17/2022 11:09 AM - Eric Faulhaber

Igor Skornyakov wrote:

> Please note also that I decided do not add underscore to the names of UDFs w/o argument. So it should be getfwdversion, not getfwdversion_. Is it OK?
> Thank you.

Yes, no need to augment names of functions with no arguments.

#69 - 08/17/2022 11:12 AM - Igor Skornyakov

Please note however, that adding "signature" suffix to the UDF names increases their total number by ~50%.
For example the number of MariaDB UDFs is 321 vs 214 in PostgreSQL.
The number of UDFs can be significantly reduces using a more sophisticated logic and using the "natural" datatypes types casting supported by the database. (For example the is not need to have UDFs with _I and _L sufixes id the logic is the same, like for e.g. eq.

#70 - 08/17/2022 11:16 AM - Greg Shah

> Devil's advocate point: there already are significant differences in syntax and builtin function names across database vendors, which any hand-written SQL would need to take into account.

I see no reason to make this problem worse, which we are doing.

> The UDFs we are porting here represent legacy, 4GL builtin functions. They exist only to support references to those existing, legacy functions in converted WHERE clauses, but one would not choose to use these in hand-written SQL. Applications which have some existing dependency on

> hand-written SQL external to the 4GL code already have had to do without them until now.

It is highly likely that over time some amount of the converted code will be migrated to pure Java and the queries to SQL. Considering the extensive usage of these functions, it is unlikely that they will be removed across the board.

> Appending some signature code to a base function name to represent overloaded signatures for these legacy functions is a somewhat ugly compromise we use to work around databases which don't support function overloading naturally. I've been trying to avoid exposing this through the converted application, because I think seeing todate(...) in a converted WHERE clause, for example, is a lot more appealing and less confusing than seeing todate_LII(...).

Why would this emit into the converted code? Our converted code should not be dialect-specific today. Don't we remap the names in the some query preprocessing step where we rewrite FQL as SQL?

The consistency is more important than the specific names that can be achieved in PostgreSQL. I think it is a net-negative to have this differentiation. It adds friction, fragility and pain points with very little value in response.

#71 - 08/17/2022 11:53 AM - Ovidiu Maxiniuc

Igor Skornyakov wrote:

> I've generated templates for MariaDB UDFs from PostgreSQL ones. This is why we have no e.g. eq_LI UDF. only eq_LL and eq_II.

I think the best would be to scan for @HQLFunction annotation in the list of public static methods from com.goldencode.p2j.persist.pl.Functions and Operators. From my POV, we can drop the implicit widening types, in your example only the eq_LL should be kept.

> Please note also that I decided do not add underscore to the names of UDFs w/o argument. So it should be getfwdversion, not getfwdversion_. Is it OK?

Yes, no problem, I just need to add it lately, conditionally when constructing the SQL name in a StringBuilder.

> fwdsession UDF does not exist for PostgreSQL. Actually I do not see how it can be implemented in a native mode.

Based on your comment in javadoc: NB: this function is for embedded H2 only!, this is not necessary.

Greg Shah wrote:

> If I understand correctly, different database "dialects" (in FWD) will have different function names for the same 4GL built-in function. Is that right?
>
> - PostgreSQL uses overloading and schema names to qualify functions, with differences between Java and native SQL functions

- H2 uses Java functions, I'm not sure how the naming works
- SQLServer has no overloading so there is some signature string appended
- MariaDB has no overloading so there is some signature string appended

I think this reflects the evolution of the code relatively to supported dialects. I think the first implementation was for PSQL. There were no issues here.

At the time H2 was added, the issue aroused and had to be handled somehow. An index was prefixed so the SQL names were nor unique. But this dialect was initially targetted to _temp database where the functions were redefined fresh for each execution. It really did not matter how these were decorated and if they differ from one execution to another, they were always registered with and used only during one process. When H2 started to be used as a more convenient solution for permanent databases (no need to install and configure PSQL) we had to make sure the names were consistent from database setup to each execution because function_K might produce an incorrect value or failed to executed because of a syntax error. In such event, the UDFs had to be redefined for each database. Luckily, the Functions and Operators were really stable by now and we never encountered such issues. At least none I know of.

Then MS SQL Server was added. I really did not like the solution for H2 and, after some experiments I came up with the signature suffix I mentioned in note-33 (driven by SQLServerHelper.getSqlDecoration()). My main concern was a possible collision with a table/column name. But this was avoided because the functions were defined in a dedicated namespace and each usage of these UDF required that they are prefixed by dbo..

The latest addition is MariaDb. The problem is explained above.

Should we unify all these solutions? My answer is definitely YES! And I think this should be done at the very low level. The Functions and Operators should expose them directly decorated, so that HQLPreprocessor to replace directly the right UDF name instead of using the lookup table, increasing performance and decreasing memory usage a bit, at the same time. But this method is quite radical: the result must be heavily tested and the existing databases to be ported to UDFs using the new naming schema. As a suggestion, I think it would be better to decide on a name casing (all lowercase maybe?) - it should not really matter because SQL is case-insensitive but we would have a more aesthetic looking generated code.

#72 - 08/17/2022 11:59 AM - Ovidiu Maxiniuc

Greg Shah wrote:

> Why would this emit into the converted code? Our converted code should not be dialect-specific today. Don't we remap the names in the some query preprocessing step where we rewrite FQL as SQL?

Yes, the generated FQL code is dialect-independent in generated classes, but then the HQLPreprocessor will convert to an intermediary dialect-specific FQL before being converted to SQL.

> The consistency is more important than the specific names that can be achieved in PostgreSQL. I think it is a net-negative to have this differentiation. It adds friction, fragility and pain points with very little value in response.

This is done automatically and transparent. I agree to consistency, as noted above.

#73 - 08/17/2022 12:03 PM - Greg Shah

> I think it would be better to decide on a name casing (all lowercase maybe?)

+1 for all lowercase names.

#74 - 08/17/2022 12:11 PM - Igor Skornyakov

Ovidiu Maxiniuc wrote:

> I think the best would be to scan for @HQLFunction annotation in the list of public static methods from com.goldencode.p2j.persist.pl.Functions and Operators. From my POV, we can drop the implicit widening types, in your example only the eq_LL should be kept.

Well, this is a possible option. It is a natural choice since UDFs a 4GL functions' counterparts which are also used in Java code,
Please note however, that some SQL UDFs contain additional arguments for injected values of the SESSION attributes. I'm also not sure that we can always rely on implicit type widening rules and that these rules are the same for all databases. Maybe @HQLFunction annotation should contain some metadata are additional annotation(s) should be used.
Anyway, I think that it is a little late to change the approach.

#75 - 08/17/2022 01:26 PM - Ovidiu Maxiniuc

Igor Skornyakov wrote:

> Well, this is a possible option. It is a natural choice since UDFs a 4GL functions' counterparts which are also used in Java code,
> Please note however, that some SQL UDFs contain additional arguments for injected values of the SESSION attributes. I'm also not sure that we can always rely on implicit type widening rules and that these rules are the same for all databases. Maybe @HQLFunction annotation should contain some metadata are additional annotation(s) should be used.

I meant only the function name, not the full statement. For example, if a case-insensitive character literal occur it is already uppercased, but when a case-sensitive character literal is used in a statement, the casing must me preserved.

> Anyway, I think that it is a little late to change the approach.

I am not sure what you refer to. The @HQLFunction is maintain by GC, we can add any attributes we need.

#76 - 08/17/2022 01:40 PM - Igor Skornyakov

Ovidiu Maxiniuc wrote:

> > Anyway, I think that it is a little late to change the approach.
>
> I am not sure what you refer to. The @HQLFunction is maintain by GC, we can add any attributes we need.

I mean that at this moment most of UDFs are already ported. The approach I've used helps a lot providing a PostgerSQL UDF code as a comment. I cannot imagine how to use the new approach without significant new efforts.

#77 - 08/17/2022 01:45 PM - Ovidiu Maxiniuc

Igor Skornyakov wrote:

> Added placeholder for the error handling UDFs in 3821c/14169.
>
> Please note that some native UDFs (such as totimestring) are not used directly by FWD code but are called from other UDFs.
> contains UDFs are used only with H2. For PostgreSQL and MariaDB word tables are used to support CONTAINS operator.
> I will add additional version of the missing UDFs.

I encountered some issues with word table DDLs. I will check the syntax to see what is wrong and how can it be fixed.

> How can I create a diff file from #6628-60 to check that I have not missed something?

I used meld to compare two text captions:

1. as the 'file name' suggests, I executed a select SPECIFIC_NAME from information_schema.routines on the database, after running the UDF import;
2. when debugging the FWD server, just before the primary database using MariaDb is initialized and preparePermanentDatabase() is about to be called, I cleared the HQLPreprocessor.overloadedFunctions. After the preparePermanentDatabase() is done, I sorted and printed the values from overloadedFunctions, like this:

```
System.out.println(new TreeSet(HQLPreprocessor.overloadedFunctions.values()))
```

Finally, exported the differences from meld as a patch.

Now, as I wrote the text above, I noticed that the UDFs are defined once for **ALL** database of a MariaDb server instance. This means that:

- executing the UDF import once is sufficient for all databases;
- executing the import in different databases will overwrite the previous. This should be no problem because the UDFs are declared using CREATE OR REPLACE FUNCTION .... It is only a temporary waste of resources. That is - if the definitions of UDF remain unchanged;
- There could be a problem is different databases use different version of FWD/UDFs. Some results may be incorrect if the original UDFs were overwritten.

#78 - 08/17/2022 01:46 PM - Igor Skornyakov

Greg Shah wrote:

> > I think it would be better to decide on a name casing (all lowercase maybe?)
>
> +1 for all lowercase names.

Suffixes converted to lowercase in 3821c/14174.

#79 - 08/17/2022 01:53 PM - Igor Skornyakov

Ovidiu Maxiniuc wrote:

> Now, as I wrote the text above, I noticed that the UDFs are defined once for **ALL** database of a MariaDb server instance. This means that:
>
> - executing the UDF import once is sufficient for all databases;
> - executing the import in different databases will overwrite the previous. This should be no problem because the UDFs are declared using CREATE OR REPLACE FUNCTION .... It is only a temporary waste of resources. That is - if the definitions of UDF remain unchanged;
> - There could be a problem is different databases use different version of FWD/UDFs. Some results may be incorrect if the original UDFs were overwritten.

I don't think that it is really a problem. I have two MariaDB databases - one is 'hotel' and another one is 'udf'. Both have two version of UDFs, And these UDFs **are** different. I understand that UDFs in different databases are actually in different namesspaces. In my case caps_T in udf is actually udf.caps_I and @caps_i in hotel is hotel.caps_i

#80 - 08/17/2022 01:57 PM - Igor Skornyakov

Ovidiu Maxiniuc wrote:

> I used meld to compare two text captions:
>
> 1. as the 'file name' suggests, I executed a select SPECIFIC_NAME from information_schema.routines on the database, after running the UDF import;
> 2. when debugging the FWD server, just before the primary database using MariaDb is initialized and preparePermanentDatabase() is about to be called, I cleared the HQLPreprocessor.overloadedFunctions. After the preparePermanentDatabase() is done, I sorted and printed the values from overloadedFunctions, like this:
> [...]

Finally, exported the differences from meld as a patch.

Thank you. This looks complicated (I mean HQL part) . Can you please check again since I understand that you already have a list based of HQL? Or please provide this list.
Thank you.

#81 - 08/17/2022 04:04 PM - Ovidiu Maxiniuc

Please expand to see the full list of expected UDF in FWDthe full list of expected UDF in FWD

```
addinterval_dit
addinterval_dlt
addinterval_sit
addinterval_slt
begins_tt
begins_ttb
caps_t
checkerror_bb
concat_tt
contains_tt
contains_ttb
datespan_dd
divide_ii
divide_il
divide_in
divide_li
divide_ll
divide_ln
divide_ni
divide_nl
divide_nn
entryin_it
entryin_itt
entryin_lt
entryin_ltt
eq_bb
eq_dd
eq_ii
eq_il
eq_in
eq_li
eq_ll
eq_ln
eq_ni
eq_nl
eq_nn
eq_ss
eq_tt
fill_ti
fill_tl
fwdsession
getday_d
getday_s
getfwdversion
getinterval_ddt
getinterval_dst
getinterval_sdt
getinterval_sst
getmonth_d
getmonth_s
getmtime
getmtime_s
gettimezone
getweekday_d
getweekday_s
```

```
getyear_d
getyear_s
gt_bb
gt_dd
gt_ii
gt_il
gt_in
gt_li
gt_ll
gt_ln
gt_ni
gt_nl
gt_nn
gt_ss
gt_tt
gte_bb
gte_dd
gte_ii
gte_il
gte_in
gte_li
gte_ll
gte_ln
gte_ni
gte_nl
gte_nn
gte_ss
gte_tt
indexof_tt
indexof_tti
indexof_ttib
indexof_ttl
indexof_ttlb
indexof_ttn
indexof_ttnb
initerror_b
lengthof_t
lengthof_tt
lookup_tt
lookup_ttt
lookup_tttb
lt_bb
lt_dd
lt_ii
lt_il
lt_in
lt_li
lt_ll
lt_ln
lt_ni
lt_nl
lt_nn
lt_ss
lt_tt
lte_bb
lte_dd
lte_ii
lte_il
lte_in
lte_li
lte_ll
lte_ln
lte_ni
lte_nl
lte_nn
lte_ss
lte_tt
ltrimws_t
ltrimws_tt
matches_tt
matches_ttb
matches_ttbb
matcheslist_tt
maximum_dd
maximum_ii
```

```
maximum_il
maximum_in
maximum_li
maximum_ll
maximum_ln
maximum_ni
maximum_nl
maximum_nn
maximum_ss
maximum_tt
minimum_dd
minimum_ii
minimum_il
minimum_in
minimum_li
minimum_ll
minimum_ln
minimum_ni
minimum_nl
minimum_nn
minimum_ss
minimum_tt
minus_di
minus_dl
minus_dn
minus_ii
minus_il
minus_in
minus_li
minus_ll
minus_ln
minus_ni
minus_nl
minus_nn
minusmillis_si
minusmillis_sl
minusmillis_sn
modulo_ii
modulo_il
modulo_in
modulo_li
modulo_ll
modulo_ln
modulo_ni
modulo_nl
modulo_nn
multiply_ii
multiply_il
multiply_in
multiply_li
multiply_ll
multiply_ln
multiply_ni
multiply_nl
multiply_nn
ne_bb
ne_dd
ne_ii
ne_il
ne_in
ne_li
ne_ll
ne_ln
ne_ni
ne_nl
ne_nn
ne_ss
ne_tt
numentries_t
numentries_tt
plus_di
plus_dl
plus_dn
plus_id
plus_ii
```

```
plus_il
plus_in
plus_ld
plus_li
plus_ll
plus_ln
plus_nd
plus_ni
plus_nl
plus_nn
plusmillis_ns
plusmillis_si
plusmillis_sl
plusmillis_sn
replacetext_ttt
reportprecisionscale_n
rounddec_n
rounddec_ni
rounddec_nl
substringof_ti
substringof_tii
substringof_tiit
substringof_til
substringof_tilt
substringof_tl
substringof_tli
substringof_tlit
substringof_tll
substringof_tllt
timespan_ss
tochr_i
tochr_l
todate_i
todate_iii
todate_iil
todate_ili
todate_ill
todate_l
todate_lii
todate_lil
todate_lli
todate_lll
todate_s
todate_t
todate_tti
todatetime_d
todatetime_di
todatetime_dl
todatetime_iiiii
todatetime_iiiiii
todatetime_iiiiiii
todatetime_lllll
todatetime_llllll
todatetime_lllllll
todatetime_t
todatetime_tti
todec_b
todec_d
todec_i
todec_l
todec_n
todec_ni
todec_nl
todec_t
toint64_b
toint64_d
toint64_i
toint64_l
toint64_n
toint64_t
toint_b
toint_d
toint_i
toint_l
toint_n
```

```
toint_t
tological_t
tological_tt
torowid_t
tostring_b
tostring_bt
tostring_d
tostring_dt
tostring_dtt
tostring_i
tostring_it
tostring_l
tostring_lt
tostring_ltt
tostring_n
tostring_nt
tostring_s
tostring_st
tostring_stt
tostring_stti
tostring_t
tostring_tt
trimws_t
trimws_tt
words_t
words_tb
words_tbb
```

Additionally, here are only the differences:

```
--- <functions registered into DB (select SPECIFIC_NAME from information_schema.routines)>
+++ <functions expected by FQL>
-checkerror_BB
+checkerror_bb
+contains_tt
+contains_ttb
-error_handler_active
+fwdsession
-indexof_ttb
-initerror_B
+initerror_b
-lookup_ttb
-matches__ttbb
-matcheslist__tt
-parse_time_format_t
-regex1_tb
-regex_t
-regex_tb
-timetostring_mti
-time_format_regexp_
-tostring_lti
-tostring_mt
-tostring_mti
-tostring_mtt
-tostring__tt
-totimestring_lti
```

Note that the FWD revision used in this listing is not committed yet.

#82 - 08/17/2022 04:06 PM - Igor Skornyakov

Ovidiu Maxiniuc wrote:

> Please expand to see {{collapse(the full list of expected UDF in FWD)
> [...]
> }}
>
> Additionally, here are only the differences:
> [...]
>
> Note that the FWD revision used in this listing is not committed yet.

Thank you, Ovidiu. It looks that now no UDFs are missed.

#83 - 08/19/2022 09:58 AM - Eric Faulhaber

What is the revised total number of UDFs for this dialect, and how many currently have a first pass implementation? I know you addressed the simpler ones first, so I expect the pace to be slowing, but I'm still curious. Thanks.

#84 - 08/19/2022 11:27 AM - Igor Skornyakov

Eric Faulhaber wrote:

> What is the revised total number of UDFs for this dialect, and how many currently have a first pass implementation? I know you addressed the simpler ones first, so I expect the pace to be slowing, but I'm still curious. Thanks.

At this moment there are 316 MariaDB UDFs, 275 of them are fully implemented (not extensively tested yet), 41 are just placeholders.
I hope that today I will commit ported matches, matcheslist and lookup.

#85 - 08/19/2022 11:40 AM - Igor Skornyakov

Update.
Here is the complete list of MariaDB UDFs to be implemented:

```
totimestring_lti(v BIGINT, fmt TEXT, tz INTEGER)
entryin_ltt(idx BIGINT, list TEXT, delim TEXT)
lookup_tttb(entry TEXT, list TEXT, delim TEXT, cs BOOLEAN)
matches_ttbb(op TEXT, pattern TEXT, casesens BOOLEAN, windows BOOLEAN)
matcheslist_tt(list TEXT, item TEXT)
numentries_t(list TEXT)
numentries_tt(list TEXT, delim TEXT)
reportprecisionscale_n(x DECIMAL(50, 10))
```

```
substringof_tiit(s TEXT, pos INTEGER, len INTEGER, stype TEXT)
timetostring_mti(t TIME, fmt TEXT, tz INTEGER)
todate_tti(s TEXT, fmt TEXT, wyear INTEGER)
todate_t(s TEXT)
todatetime_tti(s TEXT, fmt TEXT, wyear INTEGER)
todec_t(v TEXT)
toint_t(v TEXT)
toint64_t(v TEXT)
tological_tt(v TEXT, fmt TEXT)
torowid_t(v TEXT)
tostring_bt(v BOOLEAN, fmt TEXT)
tostring_mti(t TIME, fmt TEXT, tz INTEGER)
tostring_tt(v TEXT, fmt TEXT)
tostring__tt(v TEXT, fmt TEXT)
tostring_dtt(dt DATE, fmt TEXT, sfmt TEXT)
tostring_stti(ts TIMESTAMP, fmt TEXT, sfmt TEXT, tz INTEGER)
tostring_stt(ts TIMESTAMP, fmt TEXT, sfmt TEXT)
tostring_nt(v DECIMAL(50, 10), fmt TEXT)
tostring_mtt(t TIME, fmt TEXT, sep TEXT)
```

#86 - 08/19/2022 01:58 PM - Igor Skornyakov

Does anybody know a replacement of PostgreSQL RAISE NOTICE statement for MariaDB?
I've used it for debugging complex UDFs.
Thank you.

#87 - 08/19/2022 02:54 PM - Eric Faulhaber

According to https://dev.mysql.com/doc/refman/8.0/en/signal.html, using SIGNAL with a SQLSTATE which begins with 01 signals a warning, which does not abort the current operation. It does seem to have side effects, like incrementing the warning count, though I'm not sure we care, if these signals are only enabled while developing/porting the UDFs, and commented out once they are running well. It doesn't seem to be exactly the equivalent of RAISE NOTICE, as apparently you have to do something explicit to read the warnings, if I understand correctly. I have not experimented, so this is just from Googling, FWIW.

Please let me know if this works for your purposes. Otherwise, I can reach out to people who know more...

#88 - 08/19/2022 03:00 PM - Igor Skornyakov

Eric Faulhaber wrote:

> According to https://dev.mysql.com/doc/refman/8.0/en/signal.html, using SIGNAL with a SQLSTATE which begins with 01 signals a warning, which does not abort the current operation. It does seem to have side effects, like incrementing the warning count, though I'm not sure we care, if these signals are only enabled while developing/porting the UDFs, and commented out once they are running well. It doesn't seem to be exactly the equivalent of RAISE NOTICE, as apparently you have to do something explicit to read the warnings, if I understand correctly. I have not experimented, so this is just from Googling, FWIW.

Please let me know if this works for your purposes. Otherwise, I can reach out to people who know more...

Thank you, Eric.
I've tried this, but have not found how to see the result using my database client (dbeaver). I've found a solution by creating a table and using SP which stores the message in it. It is much less convenient than RAISE NOTICE but better than nothing.
If course after debugging I will comment these log calls as I did with RAISE NOTICE.

#89 - 08/22/2022 03:42 PM - Igor Skornyakov

Ported matches, matcheslist, lookup.

Committed to 3821c/14184.

#90 - 08/23/2022 01:55 PM - Igor Skornyakov

Ported the following UDFs:

```
entryin(idx BIGINT, list TEXT, delim TEXT)
numentries(list TEXT)
numentries(list TEXT, delim TEXT)
torowid(v TEXT)
todec(v TEXT)
toint(v TEXT)
toint64(v TEXT)
reportprecisionscale(x DECIMAL(50, 10))
substringof(s TEXT, pos INTEGER, len INTEGER, stype TEXT)
```

Committed to 3821c/14191.

The following **13** UDFs are still to be ported:

```
totimestring_lti(v BIGINT, fmt TEXT, tz INTEGER)
timetostring_mti(t TIME, fmt TEXT, tz INTEGER)
todate_tti(s TEXT, fmt TEXT, wyear INTEGER)
todatetime_tti(s TEXT, fmt TEXT, wyear INTEGER)
tological_tt(v TEXT, fmt TEXT)
tostring_bt(v BOOLEAN, fmt TEXT)
tostring_mti(t TIME, fmt TEXT, tz INTEGER)
tostring_tt(v TEXT, fmt TEXT)
tostring_dtt(dt DATE, fmt TEXT, sfmt TEXT)
tostring_stti(ts TIMESTAMP, fmt TEXT, sfmt TEXT, tz INTEGER)
tostring_stt(ts TIMESTAMP, fmt TEXT, sfmt TEXT)
tostring_nt(v DECIMAL(50, 10), fmt TEXT)
tostring_mtt(t TIME, fmt TEXT, sep TEXT)
```

#91 - 08/24/2022 03:36 PM - Igor Skornyakov

Ported the following UDFs:

```
totimestring(v BIGINT, fmt TEXT, tz INTEGER)
tological(v TEXT, fmt TEXT)
```

Committed to 3821c/14196.

The following **11** UDFs are still to be ported:

```
timetostring_mti(t TIME, fmt TEXT, tz INTEGER)
todate_tti(s TEXT, fmt TEXT, wyear INTEGER)
todatetime_tti(s TEXT, fmt TEXT, wyear INTEGER)
tostring_bt(v BOOLEAN, fmt TEXT)
tostring_mti(t TIME, fmt TEXT, tz INTEGER)
tostring_tt(v TEXT, fmt TEXT)
tostring_dtt(dt DATE, fmt TEXT, sfmt TEXT)
tostring_stti(ts TIMESTAMP, fmt TEXT, sfmt TEXT, tz INTEGER)
tostring_stt(ts TIMESTAMP, fmt TEXT, sfmt TEXT)
tostring_nt(v DECIMAL(50, 10), fmt TEXT)
tostring_mtt(t TIME, fmt TEXT, sep TEXT)
```

#92 - 08/25/2022 04:08 PM - Igor Skornyakov

Ported the following UDFs:

```
todate(s TEXT, fmt TEXT, wyear INTEGER)
```

Committed to 3821c/14199.

The following **10** UDFs are still to be ported:

```
timetostring_mti(t TIME, fmt TEXT, tz INTEGER)
todatetime_tti(s TEXT, fmt TEXT, wyear INTEGER)
tostring_bt(v BOOLEAN, fmt TEXT)
tostring_mti(t TIME, fmt TEXT, tz INTEGER)
tostring_tt(v TEXT, fmt TEXT)
tostring_dtt(dt DATE, fmt TEXT, sfmt TEXT)
tostring_stti(ts TIMESTAMP, fmt TEXT, sfmt TEXT, tz INTEGER)
tostring_stt(ts TIMESTAMP, fmt TEXT, sfmt TEXT)
tostring_nt(v DECIMAL(50, 10), fmt TEXT)
tostring_mtt(t TIME, fmt TEXT, sep TEXT)
```

#93 - 08/25/2022 05:26 PM - Eric Faulhaber

Igor Skornyakov wrote:

> Ported the following UDFs:
> [...]
>
> Committed to 3821c/14199.
>
> The following **10** UDFs are still to be ported:
> [...]

Does this mean 306 of 316 Maria UDFs are implemented now? As I understand it, the 316 total number is based on the UDFs implemented for the other dialects we support, taking into consideration all of the signature variants for overloaded use. So, porting these final 10 gets us to parity with the other dialects, correct?

Did you cross-reference the list of those you are porting with the list in #6628-52? I realize not all the functions in the latter list represent functions that would need to be implemented as database server-side UDFs, but for those that could be used as such, we want to be sure they are implemented.

Can I assume that you anticipate the last 10 in your list to be the most challenging implementations? Do you know of any potentially blocking issues there?

Have you been testing the implementations against the suite of 4GL tests you created for UDF verification as you go, or is the plan to run these tests once all the UDFs are implemented?

#94 - 08/25/2022 05:41 PM - Igor Skornyakov

Eric Faulhaber wrote:

> Does this mean 306 of 316 Maria UDFs are implemented now? As I understand it, the 316 total number is based on the UDFs implemented for the other dialects we support, taking into consideration all of the signature variants for overloaded use. So, porting these final 10 gets us to parity with the other dialects, correct?

Yes, this is correct.

> Did you cross-reference the list of those you are porting with the list in #6628-52? I realize not all the functions in the latter list represent functions that would need to be implemented as database server-side UDFs, but for those that could be used as such, we want to be sure they are implemented.

I understand that not all functions from #6628-52? have UDF counterparts (neither Java no PostgreSQL). It seems that the remaining ones are already ported. In fact the non-ported ones are all about format-driven conversion from/to strings.

> Can I assume that you anticipate the last 10 in your list to be the most challenging implementations? Do you know of any potentially blocking issues there?

Yes, the remaining UDFs are most complex ones. Apart from them I see potential problems with word tables' support and error handling (guarded versions of UDFs).

> Have you been testing the implementations against the suite of 4GL tests you created for UDF verification as you go, or is the plan to run these tests once all the UDFs are implemented?

I've tested most of non-trivial UDFs, but have not run a complete compatibility test suite yet. I think that it makes sense to postpone it while all UDFs will be ported. This will save time for the test suite modification I've planned before. I also do not want to stop coding with MariaDB SQL since it is much more tricky business than with PostgreSQL and requires practice.

#95 - 08/25/2022 05:59 PM - Eric Faulhaber

Igor Skornyakov wrote:

> Eric Faulhaber wrote:
>
>> So, porting these final 10 gets us to parity with the other dialects, correct?
>
> Yes, this is correct.

Nice work!

> I understand that not all functions from #6628-52? have UDF counterparts (neither Java no PostgreSQL). It seems that the remaining ones are already ported. In fact the non-ported ones are all about format-driven conversion from/to strings.

This list comes from an application which we must support fully on MariaDB in September. Please identify those functions in the list which are not already ported (nor are already on your list to port), and which you think could be used as a UDF. If that reduced list is reasonably short, I can check the context of how they are used, so we can make sure those that need to be supported as UDFs are identified. Even if we currently don't support them in other dialects, we will need to add them for MariaDB. We can back fill the other dialects later.

> Yes, the remaining UDFs are most complex ones. Apart from them I see potential problems with word tables' support and error handling (guarded versions of UDFs).

It is important to figure out an approach on the error handling as soon as possible. I know the PostgreSQL and H2 implementations have a dependency on order of evaluation of function parameter sub-expressions. Is the concern related to this or something else? Please keep me apprised of your thoughts on this point. Thanks.

#96 - 08/25/2022 06:03 PM - Igor Skornyakov

Eric Faulhaber wrote:

> This list comes from an application which we must support fully on MariaDB in September. Please identify those functions in the list which are not already ported (nor are already on your list to port), and which you think could be used as a UDF. If that reduced list is reasonably short, I can check the context of how they are used, so we can make sure those that need to be supported as UDFs are identified. Even if we currently don't support them in other dialects, we will need to add them for MariaDB. We can back fill the other dialects later.

OK, I will do it tomorrow.

> It is important to figure out an approach on the error handling as soon as possible. I know the PostgreSQL and H2 implementations have a dependency on order of evaluation of function parameter sub-expressions. Is the concern related to this or something else? Please keep me apprised of your thoughts on this point. Thanks.

I understand this. I hope that after finishing with port I will know MariaDB SQL much better to deal with this problem.

#97 - 08/26/2022 04:01 AM - Igor Skornyakov

Eric Faulhaber wrote:

> The following is a list of 4GL builtin functions used within WHERE clauses, based on their presence in the application which first needs to run with MariaDB as the backend:
>
> 1. input()
> 2. recid()
> 3. entry()
> 4. rowid()
> 5. lookup()
> 6. string()
> 7. substring()
> 8. today()
> 9. index()
> 10. num-entries()
> 11. decimal()
> 12. int64()
> 13. trim()
> 14. r-index()
> 15. integer()

16. frame-line()
17. frame-down()
18. maximum()
19. truncate()
20. can-do()
21. chr()
22. replace()
23. valid-handle()
24. program-name()
25. ldbname()
26. caps()
27. weekday()
28. length()
29. right-trim()
30. substitute()
31. minimum()
32. available()
33. sdbname()
34. interval()
35. now()
36. userid()
37. lc()
38. pdbname()

This list was taken from a FWD Analytics report, and the order of the list represents the frequency of use of these builtin functions (most frequently used functions first). As such, it roughly represents the priority of the corresponding implementation of the UDFs for MariaDB.

However, the report only recognizes the *presence* of a builtin function in a WHERE, clause. It *does not* differentiate between a builtin function which will be executed in the FWD server as part of a query substitution parameter (i.e., once per query, before the query is executed), and one which must execute as a database server-side UDF (i.e., once per row, during query execution). The report was too voluminous, such that tracking down every use of each of these functions to filter out the ones that were only used in query substitution parameter sub-expressions was not practical. However, if you have questions about specific functions (e.g., if the implementation of some UDF seem particularly difficult and you want to know if a particular subset of functions is actually used as UDFs), please ask.

Please find the status of the UDFs in question below:

1. input() - UI, doesn't make sense at the database layer
2. recid() - Returns the unique internal identifier of the database record currently associated with the record buffer you name. No UDF is required
3. entry() - entry_<suffix> UDFs, ported
4. rowid() - Returns the unique internal identifier of the database record currently associated with the record buffer you name. No UDF is required
5. lookup()- lookup _<suffix> UDFs, ported
6. string()- Converts a value of any data type into a character value. In progress
7. substring() - substring_<suffix> UDFs, ported
8. **today()** - Returns the current system date.
9. index() -indexof_<suffix> UDFs, ported
10. num-entries() - numentries_<suffix> UDFs, ported
11. decimal() - todec_<suffix> UDFs, ported
12. int64() toint64__<suffix> UDFs, ported
13. trim() - trimws_<suffix> UDFs, ported
14. **r-index()** - Returns an INTEGER value that indicates the position of the target string within the source string. In contrast to the INDEX function, R-INDEX performs the search from right to left.
15. integer() - toint_<suffix> UDFs, ported
16. frame-line() - UI, doesn't make sense at the database layer
17. frame-down() - UI, doesn't make sense at the database layer
18. maximum() - maximum_<suffix> UDFs (only for two arguments), ported
19. truncate() -todec_n(I,i) UDFs, ported.
20. can-do() - matcheslist_<suffix> UDFs, ported
21. chr() - tochr_<suffix> UDFs, ported
22. replace() - replacetext_ttt UDF, ported
23. valid-handle() - Verifies that a handle is valid, doesn't make sense at the database layer.
24. program-name() - Returns the name of the calling program , doesn't make sense at the database layer.
25. ldbname() - Returns the logical name of a database that is currently connected, doesn't make sense at the database layer.
26. caps() - caps_t UDF, ported
27. weekday() - getweekday__<suffix> UDFs, ported
28. length() - lengthof_<suffix> UDFs, ported
29. **right-trim()** - Removes trailing white space, or other specified characters
30. **substitute()** - Returns a character string that is made up of a base string plus the substitution of arguments in the string
31. minimum() - minimum_<suffix> UDFs (only for two arguments), ported
32. available() - Returns a TRUE value if the record buffer you name contains a record and returns a FALSE value if the record buffer is empty, doesn't make sense at the database layer
33. sdbname() - Returns the logical name of this database, doesn't make sense at the database layer
34. interval() - getinterval_<suffix> UDFs, ported.

35. **now()** - Returns the current system date, time, and time zone **as a DATETIME-TZ value**. The NOW function returns the system date and time of the client or server machine that serves as the time source for applications running during the ABL session (specified by the TIME-SOURCE attribute)./
36. userid() - Returns a character string representing the user ID for the specified database connection identity, doesn't make sense at the database layer
37. **lc()** - Converts any uppercase characters in a CHARACTER or LONGCHAR expression to lowercase characters, and returns the result. The data type of the returned value matches the data type of the expression passed to the function.
38. pdbname() - Returns the physical name of a currently connected database, doesn't make sense at the database layer

The names in **bold** are the functions which do not have corresponding UDFs (for all dialects).

As we can see after finishing porting UDFs mentioned in #6628-92, we will need to add only a few trivial UDFs, but for all(?) dialects.
There are two UDFs which require additional considerations:

- now() - Returns the current system date, time, and time zone **as a DATETIME-TZ value**. The NOW function returns the system date and time of the client or server machine that serves as the time source for applications running during the ABL session (specified by the TIME-SOURCE attribute)./
- substitute() - Returns a character string that is made up of a base string plus the substitution of arguments in the string

The now UDF is supposed to return DATETIME-TZ which has no counterpart at least in MariaDB and depends on the session time source.
The substitute is a variable arguments function. AFAIK this is not supported for UDFs neither by PostgreSQL nor by MariaDB (not sure about H2)

#98 - 08/26/2022 10:53 AM - Eric Faulhaber

For the initial application (at least based on static code analysis), only r-index and right-trim are used in contexts where they would need to operate on the database server side. The other bolded functions are used in query substitution parameter expressions which would be evaluated on the FWD server before executing the query.

If a function cannot accept a parameter which could reference a column/field in the current database table (i.e., the one being queried), then it does not need to be implemented as a UDF. now and today fall into this category.

substitute could be used as a UDF, but it is not in this initial application. Again, this is based on static code analysis; we cannot tell about dynamic queries at this point. So, the varargs nature of this one may be a problem area later.

#99 - 08/26/2022 12:55 PM - Igor Skornyakov

Eric Faulhaber wrote:

> For the initial application (at least based on static code analysis), only r-index and right-trim are used in contexts where they would need to operate on the database server side. The other bolded functions are used in query substitution parameter expressions which would be evaluated on the FWD server before executing the query.

Please note that if r-index and right-trim are used on the WHERE clause of the query they are converted to the calls of UDFs with the same names which may be not good for all dialects. Also lc does not look a good name for an UDF.

#100 - 08/26/2022 03:06 PM - Igor Skornyakov

Ported the following UDFs:

```
todatetime_tti(s TEXT, fmt TEXT, wyear INTEGER)
```

Committed to 3821c/14201.

The following **9** UDFs are still to be ported:

```
timetostring_mti(t TIME, fmt TEXT, tz INTEGER)
tostring_bt(v BOOLEAN, fmt TEXT)
tostring_mti(t TIME, fmt TEXT, tz INTEGER)
tostring_tt(v TEXT, fmt TEXT)
tostring_dtt(dt DATE, fmt TEXT, sfmt TEXT)
tostring_stti(ts TIMESTAMP, fmt TEXT, sfmt TEXT, tz INTEGER)
tostring_stt(ts TIMESTAMP, fmt TEXT, sfmt TEXT)
tostring_nt(v DECIMAL(50, 10), fmt TEXT)
tostring_mtt(t TIME, fmt TEXT, sep TEXT)
```

#101 - 08/27/2022 03:01 AM - Igor Skornyakov

Eric Faulhaber wrote:

> substitute could be used as a UDF, but it is not in this initial application. Again, this is based on static code analysis; we cannot tell about dynamic queries at this point. So, the varargs nature of this one may be a problem area later.

I think that we can implement a substitute for MariaDB as a two-argument UDF with JSON array as a second argument. For PostrgreSQL the second argument can be ARRAY.
I doubt that anybody is concerned about performance when using substitute in queries.

#102 - 08/27/2022 05:15 AM - Igor Skornyakov

> I think that we can implement a substitute for MariaDB as a two-argument UDF with JSON array as a second argument. For PostrgreSQL the second argument can be ARRAY.
> I doubt that anybody is concerned about performance when using substitute in queries.

I mean something like that. The UDF is defined like this:

```
CREATE OR REPLACE FUNCTION substitute_tf(base TEXT, args JSON )
RETURNS TEXT
DETERMINISTIC
BEGIN
  ....
END;
$$
```

The SUBSTITUTE(base, arg1, arg2, ..., argn) will be converted (re-written) as

```
substitute_tf(base, json_array(arg1, arg2, ..., argn))
```

#103 - 08/27/2022 05:19 AM - Igor Skornyakov

Igor Skornyakov wrote:

> I think that we can implement a substitute for MariaDB as a two-argument UDF with JSON array as a second argument. For PostrgreSQL the second argument can be ARRAY.

Another option is to define 9 different version of the substitute UDF since the number of substitution parameters can be 1..9.

#104 - 08/27/2022 08:27 AM - Igor Skornyakov

Implemented substitute_tf UDF (see #6628-102).
Committed to 3821c/14202.

#105 - 08/27/2022 11:57 AM - Igor Skornyakov

Added rindexof_<suffix>, rtrimws_<suffix>, and lc_t UDFs corresponding to R-INDEX, RIGHT-TRIM and LC 4GL functions.

Committed to 3821c/14203.

#106 - 08/30/2022 04:07 AM - Eric Faulhaber

Igor, is MariaDB 10.8 the minimum version on which the UDFs are supported? I've been trying to get Hotel GUI going with a MariaDB 10.5 back end, but I'm hitting the following error in the UDF installation, while processing 'ant import.db':

```
sql.udfs.mariadb:
     [java] SLF4J: Class path contains multiple SLF4J bindings.
     [java] SLF4J: Found binding in [jar:file:/home/ecf/projects/samples/convert/hotel_gui_maria/deploy/lib/sl
f4j-jdk14-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
     [java] SLF4J: Found binding in [jar:file:/home/ecf/projects/samples/convert/hotel_gui_maria/deploy/lib/sl
f4j-simple-1.6.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
     [java] SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
     [java] SLF4J: Actual binding is of type [org.slf4j.impl.JDK14LoggerFactory]
     [java] Aug 30, 2022 4:01:51 AM org.mariadb.jdbc.util.log.Slf4JLogger warn
     [java] WARNING: Error: 1064-42000: You have an error in your SQL syntax; check the manual that correspond
s to your MariaDB server version for the right syntax to use near 'declare y, n, s, lv, lfmt, yv, nv text;
     [java]    declare pos, i integer;
     [java]       set ...' at line 2
     [java] Exception in thread "main" java.sql.SQLSyntaxErrorException: (conn=37) You have an error in your S
QL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near '
declare y, n, s, lv, lfmt, yv, nv text;
     [java]    declare pos, i integer;
     [java]       set ...' at line 2
     [java]     at org.mariadb.jdbc.export.ExceptionFactory.createException(ExceptionFactory.java:280)
     [java]     at org.mariadb.jdbc.export.ExceptionFactory.create(ExceptionFactory.java:368)
     [java]     at org.mariadb.jdbc.message.ClientMessage.readPacket(ClientMessage.java:137)
     [java]     at org.mariadb.jdbc.client.impl.StandardClient.readPacket(StandardClient.java:833)
     [java]     at org.mariadb.jdbc.client.impl.StandardClient.readResults(StandardClient.java:772)
     [java]     at org.mariadb.jdbc.client.impl.StandardClient.readResponse(StandardClient.java:691)
     [java]     at org.mariadb.jdbc.client.impl.StandardClient.execute(StandardClient.java:634)
     [java]     at org.mariadb.jdbc.Statement.executeInternal(Statement.java:935)
     [java]     at org.mariadb.jdbc.Statement.executeUpdate(Statement.java:917)
     [java]     at org.mariadb.jdbc.Statement.executeUpdate(Statement.java:153)
     [java]     at com.goldencode.p2j.persist.deploy.ScriptRunner.execute(ScriptRunner.java:310)
     [java]     at com.goldencode.p2j.persist.deploy.ScriptRunner.runScript(ScriptRunner.java:275)
     [java]     at com.goldencode.p2j.persist.deploy.ScriptRunner.applyScripts(ScriptRunner.java:228)
     [java]     at com.goldencode.p2j.persist.deploy.MariaDbScriptRunner.apply(MariaDbScriptRunner.java:149)
     [java]     at com.goldencode.p2j.persist.deploy.ScriptRunner.main(ScriptRunner.java:129)
```

Is this a database version issue, or just a bug? Please advise. Thanks.

#107 - 08/30/2022 04:21 AM - Igor Skornyakov

Eric Faulhaber wrote:

> Igor, is MariaDB 10.8 the minimum version on which the UDFs are supported? I've been trying to get Hotel GUI going with a MariaDB 10.5 back end, but I'm hitting the following error in the UDF installation, while processing 'ant import.db':
>
> [...]
>
> Is this a database version issue, or just a bug? Please advise. Thanks.

Eric,
I cannot say about the minimal MariaDB version. MariaDB error messages regarding SQL syntax are often more than cryptic, but in your case it doesn't look version-dependent. However, I recall that I've recently committed udfs.sql with a corrupted definition of the tological_tt. It is fixed now.
Can you please re-test with the current udfs.sql version.
Sorry for the inconvenience.

#108 - 08/30/2022 09:39 AM - Eric Faulhaber

Igor Skornyakov wrote:

> Eric Faulhaber wrote:
>
>> Igor, is MariaDB 10.8 the minimum version on which the UDFs are supported? I've been trying to get Hotel GUI going with a MariaDB 10.5 back end, but I'm hitting the following error in the UDF installation, while processing 'ant import.db':
>>
>> [...]
>>
>> Is this a database version issue, or just a bug? Please advise. Thanks.
>
> Eric,
> I cannot say about the minimal MariaDB version. MariaDB error messages regarding SQL syntax are often more than cryptic, but in your case it doesn't look version-dependent. However, I recall that I've recently committed udfs.sql with a corrupted definition of the tological_tt. It is fixed now. Can you please re-test with the current udfs.sql version.
> Sorry for the inconvenience.

I was using 3821c/14206. The latest revision is 14207, but I do not see a newer version of udfs.sql. The reported problem is in:

```
/**
 * Returns logical representation of a char variable using a format
 */
CREATE OR REPLACE FUNCTION tological_tt(v TEXT, fmt TEXT)
RETURNS BOOLEAN
BEGIN
   declare y, n, s, lv, lfmt, yv, nv text;
   declare pos, i integer;
      set lv = lower(v);
      set lfmt = lower(fmt);
      set pos = position('/' in lfmt);
      set yv = trim(trailing from if(pos = 0, lfmt, substring(lfmt from 1 for pos - 1)));
      set nv = if(pos = 0, '', substring(lfmt from pos + 1));
      for i in 1..3
      do
         set s = elt(i, 'yes', 'true', yv );
         if (v = '' and s = '') or (v <> '' and position(v in s) = 1) then
            return true;
         end if;
      end for;
      for i in 1..3
      do
         set s = elt(i, 'no', 'false', nv );
         if position(v in s) = 1 then
            return false;
         end if;
      end for;
      return null;
END;
$$
```

#109 - 08/30/2022 10:28 AM - Igor Skornyakov

Eric Faulhaber wrote:

> I was using 3821c/14206. The latest revision is 14207, but I do not see a newer version of udfs.sql. The reported problem is in:

This revision should be OK. Let me test the imports with my version of MariaDB.

#110 - 08/30/2022 10:37 AM - Igor Skornyakov

Igor Skornyakov wrote:

> This revision should be OK. Let me test the imports with my version of MariaDB.

Eric. With my version of MariaDB Ver 15.1 Distrib 10.8.3-MariaDB the hotel database import finished w/o any problems.

#111 - 08/30/2022 11:19 AM - Eric Faulhaber

Igor Skornyakov wrote:

> Igor Skornyakov wrote:
>
>> This revision should be OK. Let me test the imports with my version of MariaDB.
>
> Eric. With my version of MariaDB Ver 15.1 Distrib 10.8.3-MariaDB the hotel database import finished w/o any problems.

OK, thanks for checking. I commented out that one function and I was able to proceed with the import. However, I am having other problems running Hotel GUI with this version of MariaDB, so I think I need to update to 10.8 or later.

Since you already apparently have gotten through the conversion, database creation, and import, can you please give Hotel GUI a quick test and let me know if you see any problems (please report these in #6348 if they are not UDF related)? In addition to the usual prepare_hotel.sh and install_spawner.sh, you will need to update the database settings in directory.xml, to reflect the correct dialect, driver class, and JDBC URL:

```
              <node class="string" name="dialect">
                <node-attribute name="value" value="com.goldencode.p2j.persist.dialect.MariaDbDialect"/>
              </node>
              <node class="container" name="connection">
                <node class="string" name="driver_class">
                  <node-attribute name="value" value="org.mariadb.jdbc.Driver"/>
                </node>
                <node class="string" name="url">
                  <node-attribute name="value" value="jdbc:mysql://localhost:3306/hotel"/>
                </node>
```

Thank you.

#112 - 08/30/2022 11:40 AM - Igor Skornyakov

Eric Faulhaber wrote:

> Igor Skornyakov wrote:
>
>> Igor Skornyakov wrote:

This revision should be OK. Let me test the imports with my version of MariaDB.

Eric. With my version of MariaDB Ver 15.1 Distrib 10.8.3-MariaDB the hotel database import finished w/o any problems.

OK, thanks for checking. I commented out that one function and I was able to proceed with the import. However, I am having other problems running Hotel GUI with this version of MariaDB, so I think I need to update to 10.8 or later.

Since you already apparently have gotten through the conversion, database creation, and import, can you please give Hotel GUI a quick test and let me know if you see any problems (please report these in #6348 if they are not UDF related)? In addition to the usual prepare_hotel.sh and install_spawner.sh, you will need to update the database settings in directory.xml, to reflect the correct dialect, driver class, and JDBC URL:

[...]

Thank you.

Eric,
I'm not familiar with the Hotel app functionality, but with a quick test (which included adding reservation) I've not noticed any problems.

#113 - 08/31/2022 02:50 PM - Igor Skornyakov

Ported the following UDFs:

```
timetostring_mti(t TIME, fmt TEXT, tz INTEGER)
```

This UDF is not used directly but is called from tostring UDFs for format-driven conversion of time/datetime

Committed to 3821c/14214.

The following **8** UDFs are still to be ported:

```
tostring_bt(v BOOLEAN, fmt TEXT)
tostring_mti(t TIME, fmt TEXT, tz INTEGER)
tostring_tt(v TEXT, fmt TEXT)
tostring_dtt(dt DATE, fmt TEXT, sfmt TEXT)
tostring_stti(ts TIMESTAMP, fmt TEXT, sfmt TEXT, tz INTEGER)
tostring_stt(ts TIMESTAMP, fmt TEXT, sfmt TEXT)
tostring_nt(v DECIMAL(50, 10), fmt TEXT)
tostring_mtt(t TIME, fmt TEXT, sep TEXT)
```

#114 - 09/02/2022 04:06 AM - Igor Skornyakov

Ported the following UDFs:

```
tostring_dtt(dt DATE, fmt TEXT, sfmt TEXT)
```

Committed to 3821c/14220.

The tostring_stt(ts TIMESTAMP, fmt TEXT, sfmt TEXT) just calls tostring_stti(ts TIMESTAMP, fmt TEXT, sfmt TEXT, tz INTEGER) with gettimezone() as additional argument.

The following **6** UDFs are still to be ported:

```
tostring_bt(v BOOLEAN, fmt TEXT)
tostring_mti(t TIME, fmt TEXT, tz INTEGER)
tostring_tt(v TEXT, fmt TEXT)
tostring_stti(ts TIMESTAMP, fmt TEXT, sfmt TEXT, tz INTEGER)
tostring_nt(v DECIMAL(50, 10), fmt TEXT)
tostring_mtt(t TIME, fmt TEXT, sep TEXT)
```

#115 - 09/02/2022 01:03 PM - Igor Skornyakov

Ported the following UDFs:

```
tostring_bt(v BOOLEAN, fmt TEXT)
tostring_mti(t TIME, fmt TEXT, tz INTEGER)
tostring_stti(ts TIMESTAMP, fmt TEXT, sfmt TEXT, tz INTEGER)
tostring_mtt(t TIME, fmt TEXT, sep TEXT)
```

Committed to 3821c/14221.

The following **2** UDFs are still to be ported:

```
tostring_tt(v TEXT, fmt TEXT)
tostring_nt(v DECIMAL(50, 10), fmt TEXT)
```

#116 - 09/05/2022 03:55 AM - Igor Skornyakov

I was thinking about error handling support ('guarded' UDFs) for MariaDB.
It looks that it can be implemented using temporary table like we do for PostgreSQL.
However, there is a problem. It looks that MariaDB does not support DROP ON COMMIT/ON COMMIT DELETE ROWS clauses in the CREATE TEMPORARY TABLE statement and does not delete rows on commit by default.
It is possible to perform a cleanup in the c3p0 ConnectionCustomizer.onCheckIn but I'm not sure that we always return connection to the pool on the query close.
Any suggestions?
Thank you.

#117 - 09/05/2022 02:22 PM - Igor Skornyakov

Ported the following UDF:

```
tostring_tt(v TEXT, fmt TEXT)
```

Committed to 3821c/14221.

The following **1** UDF is still to be ported:

```
tostring_nt(v DECIMAL(50, 10), fmt TEXT)
```

#118 - 09/07/2022 02:58 PM - Igor Skornyakov

Finished port of PostgreSQL UDFs (except ones related to error handling and word tables) to MariaDB.
Committed to 6129a/14386.

The next steps (as I understand this):

1. Running compatibility test suite
2. Added errors habling UDFs anmd 'guarded' UDFs generation (see #6628-116)
3. Port word tables' support (maybe it makes sense to implement CONTAINS UDF first?).

Is my understandong correct?
Thank you.

#119 - 09/09/2022 12:46 AM - Eric Faulhaber

Igor Skornyakov wrote:

> Finished port of PostgreSQL UDFs (except ones related to error handling and word tables) to MariaDB.
> Committed to 6129a/14386.

Yay! That's a great milestone.

> The next steps (as I understand this):
> 1. Running compatibility test suite

Yes.

> 2. Added errors habling UDFs anmd 'guarded' UDFs generation (see #6628-116)

Yes.

> 3. Port word tables' support (maybe it makes sense to implement CONTAINS UDF first?).

This will have to be deferred. Without it, I realize we can't really consider the MariaDB support to be fully complete and available for general-purpose use. However, it is not needed for the first deployment and we have other tasks that need attention.

Is there any value in implementing a CONTAINS UDF? As I recall, we can't fully support the feature in a UDF. Or is it only that the data can't be pre-processed to make the lookup efficient? Either way, I'd prefer a proper implementation over a UDF.

Please open a separate feature issue for word table support (well, word index support generally) for MariaDB.

#120 - 09/09/2022 01:40 AM - Igor Skornyakov

Eric Faulhaber wrote:

> Is there any value in implementing a CONTAINS UDF? As I recall, we can't fully support the feature in a UDF. Or is it only that the data can't be pre-processed to make the lookup efficient? Either way, I'd prefer a proper implementation over a UDF.

Of course, CONTAINS UDFs will be much less efficient than word tables, but it can be implemented in a day or two and at least our support for the CONTAINS operator will functionally complete.

#121 - 09/09/2022 01:44 AM - Igor Skornyakov

- Related to Bug #6741: Word index support for MariaDB. added

#122 - 09/13/2022 10:09 AM - Eric Faulhaber

Any issues coming out of testing? Any headway on the error handling solution? Thanks.

#123 - 09/13/2022 10:30 AM - Igor Skornyakov

Eric Faulhaber wrote:

> Any issues coming out of testing? Any headway on the error handling solution? Thanks.

Nothing unusual. Fixing revealed errors one by one.

I was not thinking about error handling recently. I still think that it can be done almost like for PostgreSQL if we will find a good solution for the problem described in #6628-116.

#124 - 09/14/2022 05:08 AM - Igor Skornyakov

There is a problem with creation of SQLWarning with MariaDB. Unlike PostgreSQL RAISE NOTICE, SIGNAL SQLSTATE '01XXX' does not generate SQLWarning.
This affects only ERROR-STATUS:NUM-MESSAGES (will be zero instead of 1 with MariaDB).
Will try to find a workaround later.

#125 - 09/14/2022 02:35 PM - Igor Skornyakov

In the converted queries there are no 'signature suffix' for indexOf and lookup.

Ovidiu, I undestand that the fix is simple. Can you please give me a clue where to look?
Thank you.

#126 - 09/14/2022 03:29 PM - Ovidiu Maxiniuc

I do not understand. There is nothing particular to those functions. I did a quick test and the functions are correctly registered for overloading:

```
registerFunction    lookup    ->    lookup_tt
registerFunction    lookup    ->    lookup_ttt
registerFunction    lookup    ->    lookup_tttb
```

If you mean the FQL code from generated WHERE predicate like:

```
new FindQuery(book, "lookup(upper(book.publisher), upper(book.isbn)) = 0", null, "book.bookId asc")
```

then this is correct. The functions are not decorated at conversion time. The FQL is dialect independent. We decided to decorate them all using the same suffixes to avoid overloading issues but this happens only at runtime, in HQL/FQLPreprocessor. When a UDF function/operator is encountered, it is manuallyOverload()-ed meaning it is renamed to associated name. In this case, the dialect-dependent preprocessed FQL is:

```
"lookup_tt(upper(rtrim(book.publisher)), upper(rtrim(book.isbn))) = 0"
```

#127 - 09/14/2022 03:33 PM - Igor Skornyakov

Ovidiu Maxiniuc wrote:

> I do not understand. There is nothing particular to those functions. I did a quick test and the functions are correctly registered for overloading:
> [...]
>
> If you mean the FQL code from generated WHERE predicate like:
> [...]
> then this is correct. The functions are not decorated at conversion time. The FQL is dialect independent. We decided to decorate them all using the same suffixes to avoid overloading issues but this happens only at runtime, in HQL/FQLPreprocessor. When a UDF function/operator is encountered, it is manuallyOverload()-ed meaning it is renamed to associated name. In this case, the dialect-dependent preprocessed FQL is:
> [...]

I see. Howver, I see the following exceptions in the log:

```
[09/14/2022 21:29:28 GMT+03:00] (com.goldencode.p2j.persist.Persistence:WARNING) [00000007:0000001C:bogus-->lo
cal/udf_4gl_tst/primary] error loading record
select udftests.recid from Udftests__Impl__ as udftests where upper(rtrim(udftests.testName)) = '_LOOKUP2CS' a
nd rtrim(udftests.fstr1cs) = ?0 and rtrim(udftests.fstr2cs) = ?1 and lookup(udftests.fstr1cs, udftests.fstr2cs
, true) = udftests.fintResult order by udftests.id asc
[09/14/2022 21:29:28 GMT+03:00] (com.goldencode.p2j.persist.Persistence:SEVERE) [00000007:0000001C:bogus-->loc
al/udf_4gl_tst/primary] error loading record
com.goldencode.p2j.persist.PersistenceException: Error uniqueResult
Caused by: java.sql.SQLSyntaxErrorException: (conn=153) FUNCTION udf_4gl_tst.lookup does not exist
```

```
09/14/2022 21:29:28 GMT+03:00] (com.goldencode.p2j.persist.Persistence:WARNING) [00000007:0000001C:bogus-->loc
al/udf_4gl_tst/primary] error loading record
select udftests.recid from Udftests__Impl__ as udftests where upper(rtrim(udftests.testName)) = '_INDEXOF2CS'
and rtrim(udftests.fstr1cs) = ?0 and rtrim(udftests.fstr2cs) = ?1 and indexOf(udftests.fstr1cs, udftests.fstr2
cs, true) = udftests.fintResult order by udftests.id asc
[09/14/2022 21:29:28 GMT+03:00] (com.goldencode.p2j.persist.Persistence:SEVERE) [00000007:0000001C:bogus-->loc
al/udf_4gl_tst/primary] error loading record
com.goldencode.p2j.persist.PersistenceException: Error uniqueResult
Caused by: java.sql.SQLSyntaxErrorException: (conn=153) FUNCTION udf_4gl_tst.indexOf does not exist
```

I will take a closer look tomorrow.

#128 - 09/15/2022 03:43 AM - Igor Skornyakov

Igor Skornyakov wrote:

> I will take a closer look tomorrow.

Using a debugger with a breakpoint in the FqlToSqlConverter.toSQL() see the following:
fql value:

```
select udftests.recid from Udftests__Impl__ as udftests where upper(rtrim(udftests.testName)) = '_INDEXOF2CS'
and rtrim(udftests.fstr1cs) = ?0 and rtrim(udftests.fstr2cs) = ?1 and indexOf(udftests.fstr1cs, udftests.fstr2
cs, true) = udftests.fintResult order by udftests.id a
```

sql statement:

```
select
    udftests__0_.recid as col_0_0_
from
    udftests udftests__0_
where
    upper(rtrim(udftests__0_.test_name)) = '_INDEXOF2CS' and rtrim(udftests__0_.fstr1cs) = ? and rtrim(udftest
s__0_.fstr2cs) = ? and indexOf(udftests__0_.fstr1cs, udftests__0_.fstr2cs, true) = udftests__0_.fint_result
order by
    udftests__0_.id asc
 limit ?
```

As we can see no suffix was added.

#129 - 09/15/2022 04:21 AM - Igor Skornyakov

Ovidiu Maxiniuc wrote:

> I do not understand. There is nothing particular to those functions. I did a quick test and the functions are correctly registered for overloading:
> [...]
>
> If you mean the FQL code from generated WHERE predicate like:
> [...]
> then this is correct. The functions are not decorated at conversion time. The FQL is dialect independent. We decided to decorate them all using the same suffixes to avoid overloading issues but this happens only at runtime, in HQL/FQLPreprocessor. When a UDF function/operator is encountered, it is manuallyOverload()-ed meaning it is renamed to associated name. In this case, the dialect-dependent preprocessed FQL is:
> [...]

Ovidiu,
what is **not** registered are indexOf and lookup for signatures [TEXT, TEXT, BOOLEAN] (_ttb suffix). These versions are called for cases-sensitive arguments, and there are no corresponding methods in Functions class.
What is the best way to fix it? Maybe just add manually?
Thank you.

#130 - 09/15/2022 05:01 AM - Igor Skornyakov

I consistently see the following log entries at the end of my UDFs' compatibility test:

```
[09/15/2022 11:57:23 GMT+03:00] (com.goldencode.p2j.persist.lock.InMemoryLockManager:WARNING) [00000007:000000
1D:bogus] --> local/udf_4gl_tst/primary:  cleaning up 161 leaked record lock(s) for exiting context ({udftests
={udftests:616=SHARE [bogus:00000007], udftests:871=SHARE [bogus:00000007], udftests:372=SHARE [bogus:00000007
], udftests:883=SHARE [bogus:00000007], udftests:370=SHARE [bogus:00000007], udftests:626=SHARE [bogus:0000000
7], udftests:369=SHARE [bogus:00000007], udftests:367=SHARE [bogus:00000007], udftests:365=SHARE [bogus:000000
07], udftests:621=SHARE [bogus:00000007], udftests:877=SHARE [bogus:00000007], udftests:636=SHARE [bogus:00000
007], udftests:378=SHARE [bogus:00000007], udftests:889=SHARE [bogus:00000007], udftests:375=SHARE [bogus:0000
0007], udftests:631=SHARE [bogus:00000007], udftests:388=SHARE [bogus:00000007], udftests:641=SHARE [bogus:000
00007], udftests:381=SHARE [bogus:00000007], udftests:651=SHARE [bogus:00000007], udftests:391=SHARE [bogus:00
000007], udftests:646=SHARE [bogus:00000007], udftests:389=SHARE [bogus:00000007], udftests:402=SHARE [bogus:0
0000007], udftests:400=SHARE [bogus:00000007], udftests:656=SHARE [bogus:00000007], udftests:397=SHARE [bogus:
00000007], udftests:412=SHARE [bogus:00000007], udftests:410=SHARE [bogus:00000007], udftests:666=SHARE [bogus
:00000007], udftests:409=SHARE [bogus:00000007], udftests:407=SHARE [bogus:00000007], udftests:405=SHARE [bogu
s:00000007], udftests:661=SHARE [bogus:00000007], udftests:420=SHARE [bogus:00000007], udftests:676=SHARE [bog
us:00000007], udftests:931=SHARE [bogus:00000007], udftests:418=SHARE [bogus:00000007], udftests:161=SHARE [bo
gus:00000007], udftests:416=SHARE [bogus:00000007], udftests:415=SHARE [bogus:00000007], udftests:671=SHARE [b
ogus:00000007], udftests:428=SHARE [bogus:00000007], udftests:427=SHARE [bogus:00000007], udftests:426=SHARE [
bogus:00000007], udftests:425=SHARE [bogus:00000007], udftests:681=SHARE [bogus:00000007], udftests:937=SHARE 
[bogus:00000007], udftests:424=SHARE [bogus:00000007], udftests:935=SHARE [bogus:00000007], udftests:422=SHARE
 [bogus:00000007], udftests:691=SHARE [bogus:00000007], udftests:947=SHARE [bogus:00000007], udftests:434=SHAR
E [bogus:00000007], udftests:943=SHARE [bogus:00000007], udftests:430=SHARE [bogus:00000007], udftests:686=SHA
RE [bogus:00000007], udftests:443=SHARE [bogus:00000007], udftests:955=SHARE [bogus:00000007], udftests:442=SH
ARE [bogus:00000007], udftests:441=SHARE [bogus:00000007], udftests:696=SHARE [bogus:00000007], udftests:949=S
HARE [bogus:00000007], udftests:452=SHARE [bogus:00000007], udftests:451=SHARE [bogus:00000007], udftests:706=
SHARE [bogus:00000007], udftests:961=SHARE [bogus:00000007], udftests:959=SHARE [bogus:00000007], udftests:190
=SHARE [bogus:00000007], udftests:701=SHARE [bogus:00000007], udftests:460=SHARE [bogus:00000007], udftests:71
6=SHARE [bogus:00000007], udftests:459=SHARE [bogus:00000007], udftests:458=SHARE [bogus:00000007], udftests:4
55=SHARE [bogus:00000007], udftests:711=SHARE [bogus:00000007], udftests:454=SHARE [bogus:00000007], udftests:
```

```
453=SHARE [bogus:00000007], udftests:468=SHARE [bogus:00000007], udftests:980=SHARE [bogus:00000007], udftests
:467=SHARE [bogus:00000007], udftests:466=SHARE [bogus:00000007], udftests:978=SHARE [bogus:00000007], udftest
s:465=SHARE [bogus:00000007], udftests:721=SHARE [bogus:00000007], udftests:976=SHARE [bogus:00000007], udftes
ts:462=SHARE [bogus:00000007], udftests:973=SHARE [bogus:00000007], udftests:988=SHARE [bogus:00000007], udfte
sts:475=SHARE [bogus:00000007], udftests:986=SHARE [bogus:00000007], udftests:473=SHARE [bogus:00000007], udft
ests:984=SHARE [bogus:00000007], udftests:471=SHARE [bogus:00000007], udftests:727=SHARE [bogus:00000007], udf
tests:982=SHARE [bogus:00000007], udftests:469=SHARE [bogus:00000007], udftests:228=SHARE [bogus:00000007], ud
ftests:996=SHARE [bogus:00000007], udftests:739=SHARE [bogus:00000007], udftests:994=SHARE [bogus:00000007], u
dftests:481=SHARE [bogus:00000007], udftests:992=SHARE [bogus:00000007], udftests:479=SHARE [bogus:00000007], 
udftests:990=SHARE [bogus:00000007], udftests:477=SHARE [bogus:00000007], udftests:733=SHARE [bogus:00000007],
 udftests:491=SHARE [bogus:00000007], udftests:745=SHARE [bogus:00000007], udftests:1000=SHARE [bogus:00000007
], udftests:486=SHARE [bogus:00000007], udftests:998=SHARE [bogus:00000007], udftests:496=SHARE [bogus:0000000
7], udftests:751=SHARE [bogus:00000007], udftests:763=SHARE [bogus:00000007], udftests:506=SHARE [bogus:000000
07], udftests:501=SHARE [bogus:00000007], udftests:757=SHARE [bogus:00000007], udftests:516=SHARE [bogus:00000
007], udftests:3=SHARE [bogus:00000007], udftests:769=SHARE [bogus:00000007], udftests:-9223372036854775808=SH
ARE [bogus:00000007], udftests:511=SHARE [bogus:00000007], udftests:521=SHARE [bogus:00000007], udftests:775=S
HARE [bogus:00000007], udftests:531=SHARE [bogus:00000007], udftests:787=SHARE [bogus:00000007], udftests:526=
SHARE [bogus:00000007], udftests:781=SHARE [bogus:00000007], udftests:793=SHARE [bogus:00000007], udftests:536
=SHARE [bogus:00000007], udftests:546=SHARE [bogus:00000007], udftests:799=SHARE [bogus:00000007], udftests:54
1=SHARE [bogus:00000007], udftests:556=SHARE [bogus:00000007], udftests:811=SHARE [bogus:00000007], udftests:5
51=SHARE [bogus:00000007], udftests:805=SHARE [bogus:00000007], udftests:561=SHARE [bogus:00000007], udftests:
817=SHARE [bogus:00000007], udftests:571=SHARE [bogus:00000007], udftests:311=SHARE [bogus:00000007], udftests
:823=SHARE [bogus:00000007], udftests:566=SHARE [bogus:00000007], udftests:835=SHARE [bogus:00000007], udftest
s:576=SHARE [bogus:00000007], udftests:829=SHARE [bogus:00000007], udftests:586=SHARE [bogus:00000007], udftes
ts:329=SHARE [bogus:00000007], udftests:841=SHARE [bogus:00000007], udftests:581=SHARE [bogus:00000007], udfte
sts:596=SHARE [bogus:00000007], udftests:337=SHARE [bogus:00000007], udftests:591=SHARE [bogus:00000007], udft
ests:847=SHARE [bogus:00000007], udftests:859=SHARE [bogus:00000007], udftests:601=SHARE [bogus:00000007], udf
tests:853=SHARE [bogus:00000007], udftests:611=SHARE [bogus:00000007], udftests:865=SHARE [bogus:00000007], ud
ftests:606=SHARE [bogus:00000007]}}})
[09/15/2022 11:57:23 GMT+03:00] (SecurityManager:SEVERE) {00000007:0000001D:bogus} SecurityContext.cleanupWork
er did not complete properly - the following tokens are still in use: [com.goldencode.p2j.persist.meta.LockTab
leUpdater$1]
```

Is it normal?
Thank you.

#131 - 09/15/2022 05:23 AM - Igor Skornyakov

Igor Skornyakov wrote:

> what is **not** registered are indexOf and lookup for signatures [TEXT, TEXT, BOOLEAN] (_ttb suffix). These versions are called for cases-sensitive arguments, and there are no corresponding methods in Functions class.
> What is the best way to fix it? Maybe just add manually?
> Thank you.

Adding indexof(text, text, boolean) and lookup(text, text, boolean) to Functions fixes the problem.
Is it OK?
Thank you.

#132 - 09/15/2022 05:21 PM - Igor Skornyakov

I see two problems which I've already seen with native PostgreSQL UDFs:

1. We have to use eq, ne, le, ... UDFs in the where expression instead of =, <>, >, ...n if both sides of the comparision are nullable. However it looks that it is not the fact.
2. if the FIND query fails due to the exception raised by UDFs the available flag should be false but it remains true if the previous FIND was successful.

I will try to recall how these problems where resolved for PostgreSQL, but it will be great if those who remember some details of this will share his memories.
Thank you.

#133 - 09/16/2022 04:48 AM - Igor Skornyakov

Igor Skornyakov wrote:

> There is a problem with creation of SQLWarning with MariaDB. Unlike PostgreSQL RAISE NOTICE, SIGNAL SQLSTATE '01XXX' does not generate SQLWarning.
> This affects only ERROR-STATUS:NUM-MESSAGES (will be zero instead of 1 with MariaDB).
> Will try to find a workaround later.

One more observation.
If I execute the script

```
SIGNAL SQLSTATE '01FWD' SET MESSAGE_TEXT = 'regex';
show warnings;
```

using MariaDB client (dbeaver in my case) I get the result set with warning:

Level	Code	Message
Warning	1642	regex

However, if I execute

```
select torowid_t('0123');
show warnings;
```

the result set returned by show warnings is empty despite the fact that torowid_t('0123') executes the same SIGNAL SQLSTATE '01FWD' SET MESSAGE_TEXT = 'regex';

It looks that it is simply impossible to generate SQLWarning from UDF with MariaDB.

#134 - 09/16/2022 05:15 AM - Igor Skornyakov

Igor Skornyakov wrote:

> Igor Skornyakov wrote:
>
> > There is a problem with creation of SQLWarning with MariaDB. Unlike PostgreSQL RAISE NOTICE, SIGNAL SQLSTATE '01XXX' does not generate SQLWarning.
> > This affects only ERROR-STATUS:NUM-MESSAGES (will be zero instead of 1 with MariaDB).
> > Will try to find a workaround later.
>
> One more observation.
> If I execute the script
> [...]
> using MariaDB client (dbeaver in my case) I get the result set with warning:
>
> | Level | Code | Message |
> |---|---|---|
> | Warning | 1642 | regex |
>
> However, if I execute
> [...]
> the result set returned by show warnings is empty despite the fact that torowid_t('0123') executes the same SIGNAL SQLSTATE '01FWD' SET MESSAGE_TEXT = 'regex';
>
> It looks that it is simply impossible to generate SQLWarning from UDF with MariaDB.

In addition.
If we execute SIGNAL '01xxx' from **stored procedure** instead of stored function, we get SQLWarning. However, if this procedure is called from UDF, the 2SQLWarning@ is supressed.

#135 - 09/16/2022 10:59 AM - Igor Skornyakov

It looks that we have problems with conversion of case-sensitive fields for MariaDB. We need to specify case-sensitive collation for such fields. Otherwise the comparision of such fields will be incorrect. For example if fstrcs1 and fstrcs2 are character CASE-SENSITIVE and fstrcs1 = 'bbb', fstrcs2 = 'BBB' then fstrcs1 <= fstrcs2 returns true.

#136 - 09/16/2022 11:09 AM - Igor Skornyakov

Igor Skornyakov wrote:

> It looks that we have problems with conversion of case-sensitive fields for MariaDB. We need to specify case-sensitive collation for such fields. Otherwise the comparision of such fields will be incorrect. For example if fstrcs1 and fstrcs2 are character CASE-SENSITIVE and fstrcs1 = 'bbb', fstrcs2 = 'BBB' then fstrcs1 <= fstrcs2 returns true.

Another options is to create database with explicit case-sensitive collation.

#137 - 09/16/2022 05:30 PM - Ovidiu Maxiniuc

Igor Skornyakov wrote:

> Adding indexof(text, text, boolean) and lookup(text, text, boolean) to Functions fixes the problem.
> Is it OK?

So, that was the explanation: missing overloaded methods in Functions. Normally, the answer is yes. However, I do not understand what's the last boolean parameter functionality? For example, the 4GL function syntax is:

```
LOOKUP ( expression , list [ , character ] )
```

The optional 3rd parameter is a char, not a logical/boolean.

> It looks that we have problems with conversion of case-sensitive fields for MariaDB. We need to specify case-sensitive collation for such fields. Otherwise the comparision of such fields will be incorrect. For example if fstrcs1 and fstrcs2 are character CASE-SENSITIVE and fstrcs1 = 'bbb', fstrcs2 = 'BBB' then fstrcs1 <= fstrcs2 returns true.
>
> Another options is to create database with explicit case-sensitive collation.

To compare character columns in case-sensitive mode in MariaDb, use binary qualifier. Ex:

```
select binary 'abc' = 'AbC'; --> 0
select 'abc' = 'AbC'; --> 1
```

#138 - 09/17/2022 05:22 AM - Igor Skornyakov

Ovidiu Maxiniuc wrote:

> Igor Skornyakov wrote:
>
> > Adding indexof(text, text, boolean) and lookup(text, text, boolean) to Functions fixes the problem.
> > Is it OK?
>
> So, that was the explanation: missing overloaded methods in Functions. Normally, the answer is yes. However, I do not understand what's the last boolean parameter functionality? For example, the 4GL function syntax is:[...]The optional 3rd parameter is a char, not a logical/boolean.

For PostgreSQL/MariaDB UDFs there is also lookup(text, text, boolean) and it is really used. There was no such UDF for Java and I've added it. The same about index.

> > It looks that we have problems with conversion of case-sensitive fields for MariaDB. We need to specify case-sensitive collation for such fields. Otherwise the comparision of such fields will be incorrect. For example if fstrcs1 and fstrcs2 are character CASE-SENSITIVE and fstrcs1 = 'bbb', fstrcs2 = 'BBB' then fstrcs1 <= fstrcs2 returns true.
> >
> > Another options is to create database with explicit case-sensitive collation.
>
> To compare character columns in case-sensitive mode in MariaDb, use binary qualifier. Ex:
> [...]

I understand that it requires changes in conversion. We do not need upper call for case-insentitive comparisions and use binary for case-sensitive ones.
Is is what you mean? I think this is not a simple change.
See also #6628-132.
Thank you.

#139 - 09/18/2022 05:32 AM - Igor Skornyakov

More about collation.
At this moment import.db creates MariaDB database with DEFAULT CHARACTER SET utf8mb4 DEFAULT COLLATE utf8mb4_general_ci. If **before data import** we change database settings to DEFAULT CHARACTER SET latin1 DEFAULT COLLATE latin1_general_cs the comparition of fields works as expected, **but** for 'bbb' = 'BBB' still returns true.
Moreover 'bbb' collate latin1_general_cs = 'BBB' is rejected with 'SQL Error [1253] [42000]: (conn=114) COLLATION 'latin1_general_cs' is not valid for CHARACTER SET 'utf8mb4',
binary 'bbb' = 'BBB' return false, but I'm not sure that e.g. binary 'bbb' < 'BBB' will return expected result for all character sets.
Strangely enough MariaDB supports only six case-sensitive collations: latin1_general_cs, latin2_czech_cs, cp1250_czech_cs, latin7_estonian_cs, latin7_general_cs, and cp1251_general_cs.
This looks wierd...

#140 - 09/19/2022 03:15 PM - Igor Skornyakov

Fixed injection of session attributes' values into UDF calls for MariaDB.
Committed to 6129a/14407.

#141 - 09/20/2022 05:44 AM - Igor Skornyakov

The compatibility test for matches fails for a strange reason.
A string literal 'a1b123*\{}()1' is converted to new character("a1b123*{}()1").
I think it was not seen before when I've tested PostgreSQL UDFs.
What I'm doing wrong?
Thank you.

#142 - 09/20/2022 12:38 PM - Ovidiu Maxiniuc

Igor Skornyakov wrote:

> A string literal 'a1b123*\{}()1' is converted to new character("a1b123*{}()1"). What I'm doing wrong?

Nothing. The \ is an escape character (only on Linux). ~ is the escape char for both OSs. FWD will process the literals during the conversion and simplify by dropping invalid sequences. This happened with your literal. Yet, I wonder why the curly braces were kept. You may want to use 'a1b123*~\~{~}()1' instead to be sure you get new character("a1b123*\{}()1")... ?

#143 - 09/20/2022 12:54 PM - Igor Skornyakov

Ovidiu Maxiniuc wrote:

> Igor Skornyakov wrote:
>
> > A string literal 'a1b123*\{}()1' is converted to new character("a1b123*{}()1"). What I'm doing wrong?

> Nothing. The \ is an escape character (only on Linux). ~ is the escape char for both OSs. FWD will process the literals during the conversion and simplify by dropping invalid sequences. This happened with your literal. Yet, I wonder why the curly braces were kept. You may want to use 'a1b123*~\~{~}()1' instead to be sure you get new character("a1b123*\{}()1")... ?

Ovidiu,
I compare the results of running the same test with 4GL and FWD. Indeed, 4GL test was executed on Windows machine and FWD was converted and executed on a Linux one. Is it possible to convert on Linux with \ not treated as escape character. As I wrote I think that I did it somehow when testing PostgreSQL native UDFs (the test was exactly the same).
Thank you.

#144 - 09/20/2022 03:25 PM - Ovidiu Maxiniuc

Set the OS of the sources in cfg/p2j.cfg.xml:

```
<parameter name="opsys" value="win32" />
```

or

```
<parameter name="opsys" value="linux" />
```

#145 - 09/20/2022 03:30 PM - Igor Skornyakov

Ovidiu Maxiniuc wrote:

> Set the OS of the sources in cfg/p2j.cfg.xml:
> [...]or[...]

Thank you, Ovidiu.
It helps!

#146 - 09/20/2022 03:56 PM - Greg Shah

It would be best to avoid such code in our testcases. If you really need something platform specific, then have a version for Linux and one for Windows and use preprocessor conditionals to choose the right one at conversion time. You should not expect that the person running conversion will always be able to set opsys exactly the way your code needs.

You can look at the testcases project in the library_calls/ directory for examples of code that is platform specific.

#147 - 09/20/2022 04:21 PM - Igor Skornyakov

Greg Shah wrote:

> It would be best to avoid such code in our testcases. If you really need something platform specific, then have a version for Linux and one for Windows and use preprocessor conditionals to choose the right one at conversion time. You should not expect that the person running conversion will always be able to set opsys exactly the way your code needs.
>
> You can look at the testcases project in the library_calls/ directory for examples of code that is platform specific.

I agree. But this is a single test in the test suite. I will think how to improve it.
Please note however that it is a **compatibility** test and I have access only to Windows 4GL environment.

#148 - 09/21/2022 10:39 AM - Igor Skornyakov

I've experienced the #4551-310 again (the compatibility test for lengthof fails because of the argument trimming).
I'm sure that this was already fixed (but I do not rememeber how exactly this was done).
Was the fix removed on purpose or it is a regression?
Thank you.

#149 - 09/21/2022 10:41 AM - Greg Shah

Isn't this just a consequence of the (currently) broken compatibility of using varchar as a datatype for MariaDB?

#150 - 09/21/2022 10:50 AM - Igor Skornyakov

Greg Shah wrote:

> Isn't this just a consequence of the (currently) broken compatibility of using varchar as a datatype for MariaDB?

No, it is not. Actually the problem is with isUDF(HQLAst n) method which relies on the udf. prefix in the UDF name. Should be re-worked for MariaDB.
Working in this.

#151 - 09/21/2022 02:01 PM - Igor Skornyakov

Fixed trimming UDFs string arguments for MariaDB.
Committed to 6191a/14412.

All compatibility tests except ones having UNKNOWN arguments and/or values essntially passed. To problem with UNKNOWN is related to using eq, ne, le, ... UDFs in the where expression instead of =, <>, >, ... if both sides of the comparision are nullable - #6628-132.
The following minor issues still exist.

1. todate_i(integer) and toint_d(date) do not support dates before October 15, 1582 (report error) - #6628-48.
2. I had to manually change default MariaDB database settings to DEFAULT CHARACTER SET latin1 DEFAULT COLLATE latin1_general_cs

before data import - #6628-(136,139).
3. For matches UDF test I had to set <parameter name="opsys" value="win32" /> in cfg/p2j.cfg.xml - #6628-(141-147).
4. If the FIND query fails due to the exception raised by UDFs the available flag should be false but it remains true if the previous FIND was successful. - #6628-132.
5. There is a problem with generating SQLWarning - #6628-(124,133,134). I have some thoughs how to deal with it, but is has perfomance problems.

#152 - 09/23/2022 11:11 AM - Igor Skornyakov

I've temporary restored rev. 13138 changes to where_clause_pre_prep.rules to run compatibility tests with nullable arguments/values.
There is a problem with adding 'signature suffix' for MariaDB UDFs in HQLPreprocessor.manuallyOverload method. The existing logic doesn't work when e.g at least one of the arguments is a function call.
Working now on a fix which will resolve the problem at least for eq, ne, le, ... UDFs when the type of at least one argument is known to finish testing.
However, the problem for more general use cases still exists.

#153 - 09/23/2022 12:58 PM - Igor Skornyakov

Igor Skornyakov wrote:

> I've temporary restored rev. 13138 changes to where_clause_pre_prep.rules to run compatibility tests with nullable arguments/values.
> There is a problem with adding 'signature suffix' for MariaDB UDFs in HQLPreprocessor.manuallyOverload method. The existing logic doesn't work when e.g at least one of the arguments is a function call.
> Working now on a fix which will resolve the problem at least for eq, ne, le, ... UDFs when the type of at least one argument is known to finish testing.
> However, the problem for more general use cases still exists.

It looks that a simple change in the DataTypeHelper.expressionType resolves the issue. All we need is to add

```
         case LPARENS:
            return expressionType((Aast)node.getFirstChild(), bufferMap);
```

to the switch.

#154 - 09/23/2022 01:03 PM - Eric Faulhaber

Wouldn't

```
case FUNCTION:
   ...
```

be more reliable?

#155 - 09/23/2022 01:24 PM - Igor Skornyakov

Eric Faulhaber wrote:

> Wouldn't
>
> [...]
>
> be more reliable?

case FUNCTION already exists.

#156 - 09/23/2022 01:47 PM - Eric Faulhaber

You need the data type of an argument to a UDF, where the argument itself is a UDF, correct?

Two questions:

- LPARENS can serve multiple purposes (e.g., the beginning of a UDF parameter list, grouping a subexpression to alter operator precedence, etc.). Is this implementation correct in the non-UDF parameter case(s)?
- In the UDF parameter case, won't this just return the expression type of the first parameter and ignore any others?

#157 - 09/23/2022 02:07 PM - Igor Skornyakov

Eric Faulhaber wrote:

> You need the data type of an argument to a UDF, where the argument itself is a UDF, correct?
>
> Two questions:
>
> - LPARENS can serve multiple purposes (e.g., the beginning of a UDF parameter list, grouping a subexpression to alter operator precedence, etc.). Is this implementation correct in the non-UDF parameter case(s)?
> - In the UDF parameter case, won't this just return the expression type of the first parameter and ignore any others?

Well, in my case the function HQLAst in the manuallyOverload was

```
eq [FUNCTION]:<id_unavailable> @1:101
   ( [LPARENS]:<id_unavailable> @1:104
      begins [FUNCTION]:<id_unavailable> @1:105
         upper [FUNCTION]:<id_unavailable> @1:112
            udftests [ALIAS]:<id_unavailable> @1:118
               fstr1 [PROPERTY]:<id_unavailable> @1:127
         upper [FUNCTION]:<id_unavailable> @1:135
            udftests [ALIAS]:<id_unavailable> @1:141
```

```
               fstr2 [PROPERTY]:<id_unavailable> @1:150
   udftests [ALIAS]:<id_unavailable> @1:160
      fboolResult [PROPERTY]:<id_unavailable> @1:169
```

.
The query was:

```
upper(udftests.testName) = '_BEGIN' and upper(udftests.fstr1) = ? and upper(udftests.fstr2) = ? and eq((begins
(upper(udftests.fstr1), upper(udftests.fstr2))), udftests.fboolResult)
```

And first child was LPARENS. So the actual problem was with actually with extra parenthesis.
Thank you, I will look deeper.

#158 - 09/23/2022 02:24 PM - Igor Skornyakov

That if end of the switch is

```
         case LPARENS:
            if (node.getNumImmediateChildren() == 1)
            {
               return expressionType((Aast)node.getFirstChild(), bufferMap);
            }
         default:
            return getTypeToken(tokType);
```

it is safe?
Thank you.

#159 - 09/25/2022 08:05 PM - Ovidiu Maxiniuc

Nice catch!
The fix from #6628-153 should be enough. When HQL/FQL are parsed, the LPARENS of the functions are dropped. Only the LPARENS which are used to alter the normal evaluation order are kept. They should have exactly one child.

However, if you want to make the code more robust/safe, the extra guard from #6628-153 is very fine. Just add a comment before the default: to let the reader know that a case fall-through takes place if LPARENS node has none or at least than 2 children.

#160 - 09/26/2022 03:03 PM - Igor Skornyakov

Testing MariaDB UDFs with UNKNOWN arguments/values, finished.

Committed to 6129a/14414 (except where_clause_pre_prep.rules)

#161 - 09/27/2022 11:17 AM - Eric Faulhaber

Igor, how did you ultimately deal with the error handler (initError, checkError, etc.)? Do any test cases in your suite test for errors raised in a nested CAN-FIND in a query, to make sure this is working as expected?

#162 - 09/27/2022 12:35 PM - Igor Skornyakov

Eric Faulhaber wrote:

> Igor, how did you ultimately deal with the error handler (initError, checkError, etc.)? Do any test cases in your suite test for errors raised in a nested CAN-FIND in a query, to make sure this is working as expected?

Eric,
I still have no final solution for error handling. See #6628-116.
However I think that today I've invented a trick based on combination of temporary tables, session variables and MariaDB weird syntax like

```
SELECT @row_number := @row_number + 1, name
FROM cities, (SELECT @row_number := 0) r;
```

that can help both with SQL warnings and error handling.
Of course this needs more detailed design and testing...

#163 - 01/24/2023 10:44 AM - Eric Faulhaber

Igor, is this task finished, other than implementing your idea for error handling? How much effort do you think it is to finish that? Thanks.

#164 - 01/24/2023 10:49 AM - Igor Skornyakov

Eric Faulhaber wrote:

> Igor, is this task finished, other than implementing your idea for error handling? How much effort do you think it is to finish that? Thanks.

Eric,
I think that error handling can be finished in 2-3 days.

One more thing is CONTAINS/word tables support. Here the only thing which I see so far is trigger's generation. I do not think that it will take significant efforts, however I'm not 100% that it is an only problem. An optimistic estimation is also 2-3 days.

#165 - 01/31/2023 01:32 AM - Eric Faulhaber

- % Done changed from 0 to 80

Igor Skornyakov wrote:

> I think that error handling can be finished in 2-3 days.

Please finish this when you reach a good point to pause #6444.

#166 - 01/31/2023 01:56 AM - Igor Skornyakov

Eric Faulhaber wrote:

> Igor Skornyakov wrote:
>
> > I think that error handling can be finished in 2-3 days.
>
> Please finish this when you reach a good point to pause #6444.

Sure.

#167 - 02/10/2023 10:56 AM - Igor Skornyakov

- File error.p added

I've experienced problem with CAN-FIND (6129c/14808).
When creating the query from the attached test I get the following exception:

```
[02/10/2023 18:47:14 GMT+03:00] (StandardServer.invoke:SEVERE) {00000009:00000025:bogus} Abnormal end!
java.lang.RuntimeException: invoke() of class com.goldencode.testcases.ias.Error_ and  method execute failed
        at com.goldencode.p2j.util.ControlFlowOps.invokeError(ControlFlowOps.java:8155)
        at com.goldencode.p2j.util.ControlFlowOps.invokeExternalProcedure(ControlFlowOps.java:6401)
        at com.goldencode.p2j.util.ControlFlowOps.invokeExternalProcedure(ControlFlowOps.java:6170)
        at com.goldencode.p2j.util.ControlFlowOps.invoke(ControlFlowOps.java:1276)
        at com.goldencode.p2j.util.ControlFlowOps.invoke(ControlFlowOps.java:883)
        at com.goldencode.p2j.main.StandardServer$LegacyInvoker.execute(StandardServer.java:2270)
        at com.goldencode.p2j.main.StandardServer.invoke(StandardServer.java:1729)
        at com.goldencode.p2j.main.StandardServer.standardEntry(StandardServer.java:625)
        at com.goldencode.p2j.main.StandardServerMethodAccess.invoke(Unknown Source)
        at com.goldencode.p2j.util.MethodInvoker.invoke(MethodInvoker.java:156)
        at com.goldencode.p2j.net.Dispatcher.processInbound(Dispatcher.java:784)
        at com.goldencode.p2j.net.Conversation.block(Conversation.java:422)
        at com.goldencode.p2j.net.Conversation.run(Conversation.java:232)
        at java.lang.Thread.run(Thread.java:748)
Caused by: java.util.NoSuchElementException
        at java.util.ArrayDeque.removeFirst(ArrayDeque.java:285)
        at java.util.ArrayDeque.pop(ArrayDeque.java:522)
        at com.goldencode.p2j.persist.orm.FqlToSqlConverter$2.ascent(FqlToSqlConverter.java:2195)
        at com.goldencode.ast.AnnotatedAst$1.notifyListenerLevelChanged(AnnotatedAst.java:2734)
        at com.goldencode.ast.AnnotatedAst$1.hasNext(AnnotatedAst.java:2643)
        at com.goldencode.p2j.persist.orm.FqlToSqlConverter.generateExpression(FqlToSqlConverter.java:2535)
        at com.goldencode.p2j.persist.orm.FqlToSqlConverter.generateWhere(FqlToSqlConverter.java:2088)
        at com.goldencode.p2j.persist.orm.FqlToSqlConverter.processSelect(FqlToSqlConverter.java:1418)
        at com.goldencode.p2j.persist.orm.FqlToSqlConverter.processStatement(FqlToSqlConverter.java:912)
        at com.goldencode.p2j.persist.orm.FqlToSqlConverter.generateExpression(FqlToSqlConverter.java:2525)
        at com.goldencode.p2j.persist.orm.FqlToSqlConverter.generateWhere(FqlToSqlConverter.java:2088)
        at com.goldencode.p2j.persist.orm.FqlToSqlConverter.processSelect(FqlToSqlConverter.java:1418)
        at com.goldencode.p2j.persist.orm.FqlToSqlConverter.processStatement(FqlToSqlConverter.java:912)
        at com.goldencode.p2j.persist.orm.FqlToSqlConverter.toSQL(FqlToSqlConverter.java:599)
```

Any suggestions?
Thank you.

#168 - 02/10/2023 09:38 PM - Ovidiu Maxiniuc

I tried using the call stack you provided but it seems different from 6129c/14808. The code at FqlToSqlConverter.java:2195 is disabled.
I did not use your test yet. I will come back with more info after converting and running it.

#169 - 02/12/2023 07:01 AM - Igor Skornyakov

I've encoutered a weird problem with MariaDB that makes the error handling based on initError/checkError impossible.

When I try to execute a simple query

```
select toint64_t(tostring_it(123456, '999999'))
```

I get an error:

```
SQL Error [1424] [HY000]: (conn=75) Recursive stored functions and triggers are not allowed
```

As far as I can see there is no recursion in toint64_t(tostring_it(123456, '999999')).
It is interesting that toint64_t(tostring_i(123456)) is processed OK.

Any suggestions?
Thank you.

#170 - 02/12/2023 07:45 AM - Igor Skornyakov

Igor Skornyakov wrote:

> I've encoutered a weird problem with MariaDB that makes the error handling based on initError/checkError impossible.
>
> When I try to execute a simple query
> [...]
> I get an error:
> [...]
> As far as I can see there is no recursion in toint64_t(tostring_it(123456, '999999')).
> It is interesting that toint64_t(tostring_i(123456)) is processed OK.
>
> Any suggestions?
> Thank you.

Actually this is a bug in tostring_it(v INTEGER, fmt TEXT).
Please disregard.
Sorry.

#171 - 02/12/2023 11:03 AM - Igor Skornyakov

- File error0.p added

- File MariaDbErrorHandler.diff added

Finished with error Handler support for MariaDB.
Because of #7113 it was tested with a simple test only
Please review.
Thank you.

#172 - 02/12/2023 11:03 AM - Igor Skornyakov

- Status changed from WIP to Review

#173 - 02/13/2023 08:48 AM - Greg Shah

> Because of #7113 it was tested with a simple test only

Go ahead and fix this now.

#174 - 02/13/2023 09:14 AM - Igor Skornyakov

Greg Shah wrote:

> > Because of #7113 it was tested with a simple test only
>
> Go ahead and fix this now.

OK, thank you.

#175 - 02/14/2023 12:28 PM - Igor Skornyakov

Igor Skornyakov wrote:

> Finished with error Handler support for MariaDB.
> Because of #7113 it was tested with a simple test only
> Please review.
> Thank you.

Please note that the changes are "pure add-on" - they just add error handler support for MariaDB. The only thing from the existed functionality that is affected is the c3p0 ConnectionCustomizer setup. I've re-tested it with PostgreSQL (the only dialect that also uses this).

Because of the MariaDB limitations fwd_user requires explicit permission to create temporary tables. This can be granted using

```
GRANT CREATE TEMPORARY TABLES ON *.* TO 'fwd_user'@'%';
```

command (mentioned in the FWDMariaDBConnectionCustomizer.java)

#176 - 02/15/2023 08:24 AM - Igor Skornyakov

Created task branch 6628a from the trunk rev.14484

#177 - 02/15/2023 09:13 AM - Igor Skornyakov

Error handler support for MariaDB committed to 6628a/14485
See also #6628-175.

Please review.
Thank you.

#178 - 06/05/2023 03:43 PM - Eric Faulhaber

Igor, as the Maria UDF implementation of the nested CAN-FIND error-handling is quite different than the PostgreSQL implementation with which I am familiar, could you please provide a high-level description of the design?

#179 - 06/05/2023 03:50 PM - Igor Skornyakov

Eric Faulhaber wrote:

> Igor, as the Maria UDF implementation of the nested CAN-FIND error-handling is quite different than the PostgreSQL implementation with which I am familiar, could you please provide a high-level description of the design?

Eric,
In fact it is very similar to the PostgreSQL implementation. I will provide a description tomorrow.
BTW: may be it makes sense to create a Wiki page both for PostgreSQL and MariaDB?
Thank you.

#180 - 06/06/2023 03:12 AM - Igor Skornyakov

Rebased 6628a to the trunk rev.14613
Pushed up to revision 14614.

#181 - 06/06/2023 05:05 AM - Igor Skornyakov

Eric Faulhaber wrote:

> Igor, as the Maria UDF implementation of the nested CAN-FIND error-handling is quite different than the PostgreSQL implementation with which I am familiar, could you please provide a high-level description of the design?

The nested CAN-FIND error handling for MariaDB is implemented in essentially the same way as for PostgreSQL - if WHERE clause contains CAN-FIND with UDFs, the FqlToSqlConverter replaces calls of these UDFs with their "guarded" versions, which intercept errors and store the error description in the temporary table temp_error_stack. However, for MariaDB the additional logic is required since MariaDB does not support ON COMMIT DROP option for temporary tables. The corresponding differences in the program logic are described below,

1. For PostgreSQL, the temp_error_stack is created in the checkerror UDFs when it is called for the first time in a query. This table is created with ON COMMIT DROP option. For MariaDB, the temp_error_stack is created once when the connection is acquired from the underlying database for incorporation into the pool in the c3p0 connection customizer. However, since there is no automatic temp_error_stack cleanup, an additional logic is required.
2. Before executing the SQL query **containing guarded UDFs calls**, the activateErrorHandler UDF is called. This UDF truncates the temp_error_stack table and sets the values of the fwd_error_ctx session variable to uuid(). The value of this variable is used to populate the fwd_error_ctx field of the temp_error_stack@in the @initError UDF. The fwd_error_ctx can be useful for troubleshooting the situations with false positive error check when the temp_error_stack was not properly cleaned up.
3. The resetErrorHandler UDF truncates the temp_error_stack table and sets the values of the fwd_error_ctx session variable to null. It is called on the SQLquery/ScrollableResults close.

#182 - 06/06/2023 07:30 AM - Greg Shah

Are you saying that every guarded UDF execution requires a UUID to be generated? That seems expensive.

#183 - 06/06/2023 07:45 AM - Igor Skornyakov

Greg Shah wrote:

> Are you saying that every guarded UDF execution requires a UUID to be generated? That seems expensive.

No. The uuid is generated (at the database side) before executing a query containing at least one guarded UDF call.

#184 - 06/06/2023 07:56 AM - Greg Shah

It still adds the cost of UUID generation to the query. This still seems costly. Why not use a sequence if you need a unique id?

#185 - 06/06/2023 07:59 AM - Igor Skornyakov

Greg Shah wrote:

> It still adds the cost of UUID generation to the query. This still seems costly. Why not use a sequence if you need a unique id?

This is a good idea. Thank you! Will be changed.

#186 - 06/06/2023 09:55 AM - Igor Skornyakov

I'm trying to re-test error handler suppoprt for MariaDB, but cannot create a database, getting the following errors on import:

```
     [java] Error: 1072-42000: Key column 'cust_num__null' doesn't exist in table
     [java] Customer.d: Error processing import data; 0 of ? record(s) successfully processed;  0 record(s) un
committed due to this error;  1 record(s) dropped.
...
     [java] Customer.d: SQL: create unique index idx__custn on customer (cust_num__null,cust_num)
     [java] Error: 1072-42000: Key column 'order_num__null' doesn't exist in table
     [java] Error: 1072-42000: Key column 'order_num__null' doesn't exist in table
```

I've tried both 'mariadb' and 'mariadblenient' for the dialect but it seems to not have any effect.
What I'm doing wrong?
Thank you.

#187 - 06/06/2023 10:29 AM - Eric Faulhaber

Igor Skornyakov wrote:

> I'm trying to re-test error handler suppoprt for MariaDB, but cannot create a database, getting the following errors on import:
> [...]
>
> I've tried both 'mariadb' and 'mariadblenient' for the dialect but it seems to not have any effect.

These look like CREATE INDEX statements for the mariadb (i.e., "strict") dialect. Those *__null generated columns don't exist in the mariadblenient dialect.

If those columns do not exist in the database, then you are trying to import using the mariadb dialect, into a database created with DDL generated by the mariadblenient dialect. The mariadb dialect would attempt to create indices this way during import.

Make sure the targetDb parameter to your import command matches the dialect used to create your database (i.e., mariadblenient when using a database created with mariadblenient; or mariadb when using a database created with mariadb).

We really should change the name of the targetDb parameter to dialect.

#188 - 06/06/2023 11:17 AM - Igor Skornyakov

Eric Faulhaber wrote:

> Igor Skornyakov wrote:
>
>> I'm trying to re-test error handler suppoprt for MariaDB, but cannot create a database, getting the following errors on import:
>> [...]
>>
>> I've tried both 'mariadb' and 'mariadblenient' for the dialect but it seems to not have any effect.
>
> These look like CREATE INDEX statements for the mariadb (i.e., "strict") dialect. Those *__null generated columns don't exist in the mariadblenient dialect.
>
> If those columns do not exist in the database, then you are trying to import using the mariadb dialect, into a database created with DDL generated by the mariadblenient dialect. The mariadb dialect would attempt to create indices this way during import.
>
> Make sure the targetDb parameter to your import command matches the dialect used to create your database (i.e., mariadblenient when using a database created with mariadblenient; or mariadb when using a database created with mariadb).
>
> We really should change the name of the targetDb parameter to dialect.

Replaced uuid() with a sequence nextval

I was unable to generate/import db in a 'strict' mode, but after adding "targetDb=${escaped.quotes}mariadblenient${escaped.quotes}" to the @import_db.xml I was able to test my changes in the 'lenient' mode.

Committed to 6628a/14615.

#189 - 09/15/2023 10:30 AM - Greg Shah

Eric: Please review.

#190 - 11/06/2023 10:58 AM - Eric Faulhaber

- % Done changed from 80 to 100

- Status changed from Review to Internal Test

Code review 6628a/14615:

The changes look good. I don't know whether Igor ever was able to test these changes, given the difficulties he documented in #6628-187.

This branch will need to be tested with an application running with the Maria lenient dialect.

#191 - 01/19/2024 01:54 AM - Eric Faulhaber

- Assignee changed from Igor Skornyakov to Alexandru Lungu

Alexandru, can you please assign this to someone on your team who has set up (or can) the large application environment with MariaDB (lenient) and regression test the changes in 6628a, revs 14614-14615 with the fwdtests?

The changes will need to be ported on top of the latest 7156b for use with that application and eventually rebased (more likely ported, as they are quite old) to latest trunk as well.

#192 - 01/19/2024 02:00 AM - Alexandru Lungu

- Assignee changed from Alexandru Lungu to Dănuț Filimon

Sure!

Danut already has the large application set-up with MariaDB lenient and can do the regression tests. Prioritize this.
I think you need to reimport MariaDB to pick up the changes in 6628a (right, Eric?)

#193 - 01/19/2024 02:04 AM - Eric Faulhaber

There are runtime changes and SQL UDF changes in this branch. Since the UDFs have changed, they need to be (re-)installed into the database. I think having a clean set of baseline data is more the reason for doing an import than the changes themselves.

#194 - 01/19/2024 02:07 AM - Dănuț Filimon

Alexandru Lungu wrote:

> Danut already has the large application set-up with MariaDB lenient and can do the regression tests. Prioritize this.

I am doing a checkout of the branch and will test it as soon as it's done.

#195 - 01/19/2024 05:26 AM - Dănuț Filimon

Added the changes from **6628a/rev.14614-14615** on top of **7156b/rev.14933**. I ran the regression tests and got between **19 and 23** failing tests (the usual number). The changes were easy to add to 7156b, with only minor conflicts.

#196 - 01/19/2024 10:10 AM - Eric Faulhaber

Thank you! Please rebase 6628a or port over the latest trunk, whichever is easier. We will need to regression test this in other environments as well, since there are changes not just to the Maria UDFs, but also to common runtime code.

#197 - 01/22/2024 03:01 AM - Dănuț Filimon

Rebased **6628a** with **trunk/rev.14934**. **6628a** is now at revision **14936**.

#198 - 01/22/2024 10:35 AM - Constantin Asofiei

Danut, did you re-generate the DDLs for the #7156 POC app, when you did the tests?

#199 - 01/23/2024 02:05 AM - Dănuț Filimon

Constantin Asofiei wrote:

> Danut, did you re-generate the DDLs for the #7156 POC app, when you did the tests?

I only did an reimport as mentioned in #6628-192 and #6628-193. In addition, I also ran ChUI regression tests (since runtime was also changed) and everything went well.

Constantin, how should we proceed?

#200 - 01/23/2024 02:06 AM - Alexandru Lungu

Danut, can you make a diff between the DDL without 6628a and the ones with 6628a?

#201 - 01/23/2024 03:00 AM - Dănuț Filimon

There is no difference.

#202 - 01/23/2024 03:19 AM - Constantin Asofiei

I've looked at 6628a rev 14936 and I think we need to solve this before merging into 7156b/trunk:

- FWDPostgreSQLConnectionCustomizer.java - this should have used 'bzr mv' but instead I think was renamed/added. Please at least fix the history entry number (002) and history text.
- FQLPreprocessor - move the history entry to be last from the list, and add the number
- guarded.sql - this is automatically generated from udfs.sql, via guard.awk. The command is awk -f guard.awk <udfs.sql >guarded.sql
  - should we add this as a step in the FWD build process?
  - what if awk is not available on the build system?
  - are we sure guarded.sql is the 'last' version built from udfs.sql?

#203 - 01/23/2024 03:35 AM - Dănuț Filimon

I fixed the history entries in **6628a/rev.14937**.

#204 - 01/26/2024 10:50 AM - Greg Shah

Can we merge this to trunk?

#205 - 01/29/2024 04:58 AM - Alexandru Lungu

Danut, any other testing required?

Beside the history entry fixes, what about the guarded.sql - was it generated from udfs.sql with guard.awk?

#206 - 01/29/2024 05:05 AM - Dănuț Filimon

Alexandru Lungu wrote:

> Danut, any other testing required?

There is no testing left that I can do. Currently ChUI and POC regression tests were done.

> Beside the history entry fixes, what about the guarded.sql - was it generated from udfs.sql with guard.awk?

As far as I know, yes. But it is not part of the FWD build process since it was manually committed.

#207 - 01/30/2024 05:39 PM - Eric Faulhaber

- Status changed from Internal Test to Merge Pending

If guarded.sql was manually committed before, please do the same with the latest version generated from udfs.sql with guard.awk (if you haven't already). Then coordinate with Greg to merge 6628a to trunk.

#208 - 01/31/2024 02:25 AM - Dănuț Filimon

guarded.sql is the latest version obtained after executing the command mentioned in #6628-202.

Greg, let me know when I can merge 6628a.

#209 - 01/31/2024 06:30 AM - Greg Shah

You can merge now.

#210 - 01/31/2024 06:53 AM - Dănuț Filimon

- Status changed from Merge Pending to Test

Branch 6628a was merged to trunk as rev.14956 and archived.

#211 - 01/31/2024 08:48 AM - Dănuț Filimon

I've made a mistake while rebasing 6628a and changed int rowSize = rowStructures.size(); to int rowSize = rowStructure.size();. How should I proceed to correct this mistake? Create 6628b, hotfix and merge it?

#212 - 01/31/2024 09:45 AM - Greg Shah

> Create 6628b, hotfix and merge it?

Yes, go ahead now.

#213 - 01/31/2024 10:34 AM - Dănuț Filimon

Branch 6628b was merged to trunk as rev.14958 and archived.

#214 - 01/31/2024 04:09 PM - Ovidiu Maxiniuc

There are still some things to fix in r14958:

- both FWDPostgreSQLConnectionCustomizer and FWDMariaDBConnectionCustomizer use slf4j instead of CentralLogger. This is strange, since the old FWDConnectionCustomizer was already upgraded;
- (cvs issue) FWDPostgreSQLConnectionCustomizer should have been moved (bzr move --auto) not deleted and then a new file added;
- FWDMariaDBConnectionCustomizer, line 83: the logger is created for wrong class;
- MariaDbLenientDialect, P2JPostgreSQLDialect, P2JSQLServer2008Dialect: all new method javadocs do not obey the coding style (@tags categories separated by an empty line and values tab-aligned);
- FqlToSqlConverter
  - line 318: naming identifiers staring with underscore (_) is, generally, not recommended. There is no solid reason to enforce this here;
  - line 419: missing empty line separator between two methods;
- the majority of changed files do not have the copyright year updated;
- SQLQuery:
  - javadoc formatting issues;
  - it seems natural that the activateErrorHandler() and resetErrorHandler() should be in pair. Yet, in some cases (scroll() method), the call to latter in finally block is disabled. OTOH, resetErrorHandler() is public, which would suggest calling it from different location in those cases, but I have not seen is called it from any other place;
  - line 673: the reason for this late review: invalid identifier makes the project build to fail (LE: seemed fixed meanwhile).

#215 - 02/01/2024 03:07 AM - Dănuț Filimon

I've looked into #6628-214 and fixed all the problems except the renamed file. I also looked in detail at all classes and also found:

- unnecessary/duplicate import statements;
- wrong logging methods being used;

Ovidiu Maxiniuc wrote:

> it seems natural that the activateErrorHandler() and resetErrorHandler() should be in pair. Yet, in some cases (scroll() method), the call to latter in finally block is disabled. OTOH, resetErrorHandler() is public, which would suggest calling it from different location in those cases, but I have not seen is called it from any other place;

resetErrorHandler() can be made private. As for the finally block, the reset is done in the hook and it should also be called when an exception is thrown.

I also noted that SQLQuery.resetErrorHandler() calls Query.getDialect(session).activateErrorHandler(session.getConnection()); and I don't think this is intended. SQLQuery.activeErrorHandler() should also use a try catch block.

#216 - 02/01/2024 04:49 AM - Dănuț Filimon

POC regression testing is ok with those changes.

#217 - 02/01/2024 11:47 AM - Eric Faulhaber

Dănuț Filimon wrote:

> POC regression testing is ok with those changes.

Danut, in which branch are these fixes? Can this still be merged today?

#218 - 02/02/2024 01:02 AM - Dănuț Filimon

The fix for #6628-214 and my mentions from #6628-215 were committed to **6628c/rev.14964**.

#219 - 02/05/2024 03:10 AM - Dănuț Filimon

Eric, do you still want 6628c to be merged?

#220 - 02/05/2024 03:14 AM - Alexandru Lungu

6628b is already in trunk and reached 7156b, so functionally we are OK in 7156b. 6628c can go in trunk without conflicting with our efforts in 7156b. As 6628c are just formatting fixes, we can go ahead with the merge on Greg's or Eric's call.

#221 - 02/05/2024 03:18 AM - Dănuț Filimon

There's a single change in SQLQuery.resetErrorHandler that should be mentioned. Instead of calling the dialect specific method resetErrorHandler, it was calling activateErrorHandler again.

#222 - 02/05/2024 10:53 AM - Eric Faulhaber

Dănuț Filimon wrote:

> There's a single change in SQLQuery.resetErrorHandler that should be mentioned. Instead of calling the dialect specific method resetErrorHandler, it was calling activateErrorHandler again.

Please explain.

#223 - 02/05/2024 10:59 AM - Dănuț Filimon

Eric Faulhaber wrote:

> Dănuț Filimon wrote:
>
>> There's a single change in SQLQuery.resetErrorHandler that should be mentioned. Instead of calling the dialect specific method resetErrorHandler, it was calling activateErrorHandler again.
>
> Please explain.

It was calling activateErrorHandler twice, once at the beginning of the method, and once in finally. In case of MariaDbLenientDialect, resetErrorHandler method would not be called.

#224 - 02/05/2024 11:56 AM - Eric Faulhaber

OK, thanks. The change seems to make logical sense, but it got me looking at this part of the code again, and I am a little concerned with the potential performance overhead being introduced. For every query execution, we are adding two dialect lookups (Query.getDialect(session)) for all dialects. Additionally, for the MariaDB dialects, we are preparing two SQL CALLs and adding two round trips to the database to activate and reset the error handler.

For the dialect lookup, I already had to make the queries dialect-aware for #6720, so when that code comes in, this should be much less expensive.

I am more concerned about the activate and reset CALLs being executed over the database connection. These are only needed when there is a nested CAN-FIND within a query (that's what the error handling in the UDFs is about). This is not the common case. The FQLPreprocessor already knows what is in the WHERE clause of every query, so we could perform these calls only when needed, which should cut out a lot of the overhead.

That being said, I haven't measured the overhead potentially added by this implementation, but we should before optimizing. I would be surprised if it is negligible, and any round-trip to the database we can avoid is a win.

Please go ahead and coordinate with Greg to merge 6628c to trunk.

Alexandru, can you suggest the best way to see if this presents a new bottleneck?

#225 - 02/05/2024 02:35 PM - Greg Shah

It sounds like these suuggested changes are simple and low risk. I don't think we have to quantify the benefit in that case. It is faster, whether or not it is measurable. If it is a large amount of change or risky, then that is a different story.

#226 - 02/06/2024 01:07 AM - Eric Faulhaber

OK, please go ahead and implement my suggestions. That is:

- augment FQLPreprocessor to detect and expose whether a nested CAN-FIND exists in the WHERE clause being (pre)processed;
- access this information in SQLQuery and/or Query;
- only invoke activeErrorHandler and resetErrorHandler when a nested CAN-FIND exists.

Please also confirm that the error handling mechanism correctly handles a CAN-FIND that is nested more than one level deep (e.g., a WHERE clause which references a table and contains a nested CAN-FIND which references a second table, which itself contains another nested CAN-FIND which references a third table).

#227 - 02/15/2024 04:23 AM - Dănuț Filimon

- % Done changed from 100 to 70

- Status changed from Test to WIP

Eric Faulhaber wrote:

> OK, please go ahead and implement my suggestions. That is:
>
> - augment FQLPreprocessor to detect and expose whether a nested CAN-FIND exists in the WHERE clause being (pre)processed;

CAN-FIND corresponds to the exists function in the were clause and can be easily identified in the FQLPreprocessor, an additional parameter for a nested CAN-FIND can be added easily.

> - access this information in SQLQuery and/or Query;

This is my main concern here, as far as I looked into, only the fql is passed to the Query/SQLQuery and the FQLPreprocessor is something that is used right when the results are executed (close to the start when the fql is assembled). This means that this new parameter needs to be passed until it reaches the Query. I suggest using FqlToSqlConverter which utilizes FQLParser and the FqlToSqlConverter.toSql method (this is available directly in Query), it also goes over the FQLAst and should be able to identify the exists function. AFAIK, the exists function is not removed/replaced while the where clause is processed. We'll pass the fql and use FQLParser to identify any nested CAN-FIND.

#228 - 02/19/2024 02:13 PM - Eric Faulhaber

Checking for exists in the FQL seems like a reasonable approach. Right now, it is only used to implement server-side CAN-FIND sub-expressions in a where clause. If that ever changes and we use exists for something else, we may need to re-visit this approach.

#229 - 02/20/2024 03:13 AM - Dănuț Filimon

I've made a few examples to check if there are other methods used and got the following:

- CAN-FIND(pt1) generated (select count()) = 1;
- CAN-FIND(FIRST pt1) generated exists();
- CAN-FIND(LAST pt1) generated exists().

The count() function is used only when CAN-FIND does not use FIRST OR LAST, but it can also be generated in other cases (found BufferImpl.count() - which I did not find an example for).

I'll try to test a large application and check what is the most often used type of CAN-FIND statement and return with the results. We can make this change if we have a larger number of CAN-FIND with LAST or FIRST and also strictly check for the (select count()) = 1 if necessary.

#230 - 02/20/2024 07:57 AM - Dănuț Filimon

After testing POC with JMXs I found out that (select count()) = 1 is used often and a lot more than with FIRST or LAST, but both cases are not used repeatedly (warmup + 5 runs resulted in <50 CAN-FIND and <10 CAN-FIND FIRST/LAST). I took each fql and tried to check if the count function is used in another context but there were none.

Files

error.p	1.29 KB	02/10/2023	Igor Skornyakov
error0.p	642 Bytes	02/12/2023	Igor Skornyakov
MariaDbErrorHandler.diff	275 KB	02/12/2023	Igor Skornyakov