# Database - Bug #2536

## improve dynamic query parser error handling

03/24/2015 02:26 AM - Eric Faulhaber

| | | | |
|---|---|---|---|
| **Status:** | New | **Start date:** | |
| **Priority:** | Low | **Due date:** | |
| **Assignee:** | Ovidiu Maxiniuc | **% Done:** | 80% |
| **Category:** | | **Estimated time:** | 0.00 hour |
| **Target version:** | | | |
| **billable:** | No | **case_num:** | |
| **vendor_id:** | GCD | **version:** | |

**Description**

### History

#### #1 - 03/24/2015 02:33 AM - Eric Faulhaber

Currently, when the P2J Progress parser encounters unexpected tokens during runtime query conversion due to invalid input, it simply dumps error messages and stack traces to stderr, and we pick these up in the server log.

Greg mentioned there is a way to configure the parser to make these errors throw an exception. This would be preferable, since it would allow us to detect a fatal error much earlier during dynamic query conversion. This would prevent unnecessary work, and more importantly, any unreliable behavior which results from attempting to continue when a fatal parsing error has occurred.

Please investigate how to configure the parser to enable this behavior. This should be set only for runtime parsing. For static conversion, we want the parser to continue working on the next file, rather than aborting a conversion batch job.

#### #2 - 03/24/2015 08:27 AM - Greg Shah

See setConsumeError() and reportError() in progress.g.

#### #3 - 03/24/2015 03:16 PM - Ovidiu Maxiniuc

*- File om_upd20150324b.zip added*

The attached update prepares the ProgressParser to throw the exception when the syntax of the processed string is not valid. However, catching it back is a little cumbersome since original RecognitionException is wrapped in a new RuntimeException.

When the new code is executed, the stack trace is still present in the console because the showConversionError() logs it with Level.SEVERE level. However, the calling method are then quickly exited with null/negative return value.

#### #4 - 03/24/2015 07:24 PM - Eric Faulhaber

Code review 0324b:

Looks fine. If you have tested successfully locally, please commit and distribute. There is no test for this in the standard regression test environment.

#### #5 - 03/25/2015 02:10 PM - Eric Faulhaber

*- Priority changed from Normal to Low*

I have applied this update and run a large set of unit tests against it. It seems to work well and we should commit it as is.

However, there's a further improvement we can make for compatibility with Progress. For static conversion, we don't really care too much about this, but at runtime, we should try to match the behavior exactly, if possible.

Our output from this failure looks like this (in one OPEN QUERY example):

```
Unable to Prepare searchQuery for : FOR EACH areachc AND areachc.dpt-cde = " " AND areachc.area-chosen = TRUE
.
Could not parse the ' FOR EACH areachc AND areachc.dpt-cde = " " AND areachc.area-chosen = TRUE ' predicate fo
r query . (-1)
```

The equivalent Progress output for the same input is:

```
Unable to Prepare searchQuery for : FOR EACH areachc AND areachc.dpt-cde = " " AND areachc.area-chosen = TRUE
.
** Unable to understand after -- "EACH areachc". (247)
PREPARE syntax is: {FOR | PRESELECT} EACH OF.. WHERE ... etc". (7324)
```

The first message looks fine and the third message appears to be boilerplate, though it will differ by query type (FIND vs. OPEN QUERY). I think now that we're catching the RecognitionException thrown by the parser, we may have enough information to recreate the second message exactly as well.

This is based on the assumption that:

- we can get the parser's column position at the point of failure;
- we can guess how far back to go in the predicate to isolate the clause of the query string to include in the "Unable to understand after -- " portion of the error message.

I suspect the second assumption will be the tricky part, as it probably is tied up in the state of the Progress parser and how it groups tokens into clauses. In other words, depending on where the error occurs, it might not be as simple as counting back 2 tokens to put this part of the message together. Also, we may not be able to get enough information out of our parser to walk back within the token stream to assemble the message.

Anyway, please have a look to see what's possible/feasible, and we can decide where to go from there.

I am dropping this issue to low priority, since the core error handling is working well, and it is just the error message output that could be made more compatible. Please add the third (boilerplate) message and add the (**) prefix to the second message in any case. These two changes should be relatively quick.

#### #6 - 03/25/2015 02:20 PM - Ovidiu Maxiniuc

I already investigated a little this issue, too, yesterday, while restoring database. I found that there are 2 kind of messages:

- must be a quoted constant or an unabbreviated,.., when the broken token is a (possible) identifier
- ** Unable to understand after + syntax on next line,  when the invalid token is a keyword

b-2:FIND-FIRST("WHERE tmp.af > 0").

```
P4GL:   tmp af must be a quoted constant or an unabbreviated, unambiguous
       buffer/field reference for buffers known to query or FIND. (7328)

P2J:   Could not parse the 'WHERE tmp.af > 0' predicate for query find_FIRST_SHARE. (-1)
exc:   NoViableAltException: unexpected token: tmp.af
       "NoViableAlt" token: ["tmp.af",<6>,line=1,col=23]
```

b-2:FIND-FIRST("WHERE And tmp.af > 0").

```
P4GL:      ** Unable to understand after -- "WHERE". (247)
       FIND METHOD syntax is: [WHERE []] [USE-INDEX ]. (10086)
```

hQuery:QUERY-PREPARE("FOR EACH book WHERE tmp.uu AND isbn = '101'").

```
P4GL:  tmp uu must be a quoted constant or an unabbreviated, unambiguous
          buffer/field reference for buffers known to query . (7328)
```

hQuery:QUERY-PREPARE("FOR EACH book WHERE AND isbn = '101'").

```
P4GL:      ** Unable to understand after -- "EACH book WHERE". (247)
       PREPARE syntax is: {FOR | PRESELECT} EACH  OF.. WHERE ... etc". (7324)

P2J:   Could not parse the 'FOR EACH book WHERE AND isbn = '101'' predicate for query . (-1)
exc:   NoViableAltException: line 1:44: unexpected token: AND
       "NoViableAlt" token: ["AND",<471>,line=1,col=44]
```

#### #7 - 03/25/2015 02:39 PM - Eric Faulhaber

It seems like we have enough information from the parser and the context of what we are doing at the time to quickly add the appropriate syntax message (line 3), but matching the other behavior looks like it could be a bit tricky, and probably is of relatively low value for most applications. Yes, we'll want to fix this ultimately, but there are higher priority issues to work on right now.

Please go ahead with adding the syntax message and we'll put the rest on hold for now.

#### #8 - 04/15/2015 01:22 PM - Ovidiu Maxiniuc

*- File om_upd20150415b.zip added*

The attached update improves message printed on fail to process query string. Depending on the failing token type a different message is printed along with the syntax suggestion, exactly as the P4GL does.

#### #9 - 04/15/2015 02:15 PM - Eric Faulhaber

Code review 0415b:

Looks good, but needs to be merged up to latest version. Assuming you've tested locally, please merge, commit, and distribute.

#### #10 - 05/18/2015 03:53 PM - Eric Faulhaber

Is there anything left to do on this issue at this point, or can it be closed?

#### #11 - 05/19/2015 03:37 AM - Ovidiu Maxiniuc

*- File om_upd20150519a.zip added*

I added a few changes to DynamicQueryHelper that refine the messages printed in console. At this moment the log is difficult to read because the long stack traces that are generated because of the bad queries generated in the client code. Also, the important information - the P4GL code was missing. I chose to display only the important information on SEVERE level and leave the long stacks on a lower level. I recommend to run with a log level superior to that and only allow the call-stacks to be printed if there is the need to.

There is still a TODO: at line 1540 in order to better match the message from P4GL but I guess this can be deferred to a later milestone.

#### #12 - 05/20/2015 12:42 PM - Eric Faulhaber

Code review 0519a:

I'm glad you made this change, I was going to ask for something like it. After you made the error messages more like Progress in the previous update, we actually lost information, because our older message was more informative than Progress. Nice to have it back.

The refactoring of the logging looks good, with one caveat: note that INFO is the default level for all loggers, so if you don't want the stack traces in the log by default (which I think is appropriate -- one typically wants to see this detail only when debugging the problem revealed by the more general error messages), you should change the logging level at line 1605 to CONFIG or lower. Assuming you manually test this change successfully, regression testing is not necessary. I would recommend making that one level change and then you can check it in.

#### #13 - 06/09/2015 01:04 PM - Ovidiu Maxiniuc

*- File om_upd20150609a.zip added*

The logging level for exceptions was lowered to FINE to keep the CONFIG level free for configuration issues.
The updated was committed to bzr as revno 10875 and was distributed by mail.

#### #14 - 06/11/2015 02:06 PM - Eric Faulhaber

*- Target version changed from Milestone 11 to 23*

*- % Done changed from 0 to 80*

#### #15 - 11/16/2016 01:21 PM - Greg Shah

*- Target version deleted (23)*

### Files

| | | | |
|---|---|---|---|
| om_upd20150324b.zip | 15.4 KB | 03/24/2015 | Ovidiu Maxiniuc |
| om_upd20150415b.zip | 16.4 KB | 04/15/2015 | Ovidiu Maxiniuc |
| om_upd20150519a.zip | 16.6 KB | 05/19/2015 | Ovidiu Maxiniuc |
| om_upd20150609a.zip | 16.6 KB | 06/09/2015 | Ovidiu Maxiniuc |