Database - Bug #5307

AdaptiveQuery goes into dynamic mode too aggressively

05/03/2021 03:58 PM - Eric Faulhaber

Status:	WIP	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Dănuț Filimon	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

Related issues:

Related to Database - Feature #6582: implement multi-table AdaptiveQuery	**WIP**
Related to Database - Support #6708: track/report at runtime when AdaptiveQue...	**Review**

History

#1 - 05/03/2021 04:48 PM - Eric Faulhaber

While implementing #3814, we used the following code in a test case:

```
DEF VAR hBuffer AS HANDLE NO-UNDO.
DEF VAR hQuery AS HANDLE NO-UNDO.
CREATE BUFFER hBuffer FOR TABLE 'customer'.
CREATE QUERY hQuery.
hQuery:SET-BUFFERS(hBuffer).
hQuery:QUERY-PREPARE('FOR EACH customer NO-LOCK').
hQuery:QUERY-OPEN().

hQuery:GET-FIRST().
```

The hQuery:GET-FIRST() converts to:

```
hQuery.unwrapQuery().getFirst();
```

getFirst eventually calls AdaptiveQuery.next(LockType), which invokes invalidateIfReferenceRowExists. This is notably NOT called from AdaptiveQuery.previous(LockType). In this test case the "reference row" exists and therefore we invalidate the preselected result set that has been fetched for the query, shifting into dynamic mode.

It is not clear to me why this shift to dynamic mode is needed. This was not part of the original AdaptiveQuery design and I am concerned it may be adversely affecting performance.

Before I go further, let's confirm something...Igor, in #3814-405, I know I asked you to post only the code snippets relevant to the issue we were discussing, and you posted the above code in #3814-406. Is there anything that has been omitted where that empty line is, between QUERY-OPEN and GET-FIRST? Because, if not, I can't see how we get into the logic which would cause the invalidation.

Pending Igor's answer to the above question...

Stanislav, this logic was added in FWD rev 10999. The bzr history entry is:

```
Fixed iteration order for non-preselect queries when the current record is updated. Various browse improvement
s and fixes including MOVE-COLUMN. Refs #2955, #2564, #3038.
```

Can you shed any additional light on why it would be necessary to invalidate the preselected results for a query as simple as the one in the above test case? I wonder if we can't be more selective about when we invalidate. The original invalidation logic did a fair amount of analysis to determine whether a record insert or update would actually impact a preselected result set before invalidating, and we need to be careful about throwing away results and making a query operate in dynamic mode, as it is orders of magnitude slower.

I temporarily added some logging to report when we were shifting to dynamic mode via the reference row mechanism and it did not appear during some basic use of a large customer application. So, my performance concern may be misplaced. Nevertheless, considering that we hit this in a very simple test case which does not seem to warrant dynamic mode, I want to understand this mechanism better and why it is needed.

#2 - 07/22/2022 10:54 AM - Eric Faulhaber

- Related to Feature #6582: implement multi-table AdaptiveQuery added

#3 - 08/25/2022 11:50 AM - Eric Faulhaber

- Related to Support #6708: track/report at runtime when AdaptiveQuery shifts into dynamic mode added

#4 - 10/26/2022 06:13 AM - Alexandru Lungu

- Assignee set to Dănuț Filimon

#5 - 11/09/2022 07:36 AM - Dănuț Filimon

- Status changed from New to WIP

I've been trying to reproduce the behavior mentioned in #5307-1 using the example provided and failed. When debugging, the invalidateIfReferenceRowExists wasn't called even once. I also wrote a few similar tests, but still couldn't manage to obtain this specific behavior.