Database - Bug #5305

dynamic function detection in RuntimeJastInterpreter

05/03/2021 03:30 PM - Eric Faulhaber

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Eric Faulhaber	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

History

#1 - 05/03/2021 03:39 PM - Eric Faulhaber

Ovidiu, while recreating a separate issue, I am getting many messages like this in the terminal:

```
Failed to detect the DYNAMIC-FUNCTION calls.
```

These are going to stderr when they really should be logged if they're important (I think they are), but aside from that...they are caused by this code in RuntimeJastInterpreter.prepare(JavaAst, boolean):

```
      if (dynamicEvaluation)
      {
         dynamicCalls = new HashMap<>();
         // scan for dynamic calls
         Iterator<Aast> it = jast.iterator();
         while (it.hasNext())
         {
            Aast next = it.next();
            if (DYNAMIC_CALL.equals(next.getText()))
            {
               dynamicCalls.put(next, NOT_EVALUATED);
            }
         }

         if (dynamicCalls.isEmpty())
         {
            System.err.println("Failed to detect the DYNAMIC-FUNCTION calls.");
         }
      }
```

DYNAMIC_CALL is defined as "ControlFlowOps.invokeDynamicFunction". This causes us to miss all of the other ControlFlowOps dynamic function method variants:

- ControlFlowOps.invokeDynamicFunctionIn
- ControlFlowOps.invokeDynamicFunctionWithMode
- ControlFlowOps.invokeDynamicFunctionInWithMode

Each of these variants is itself overloaded.

What I am actually seeing in the Java AST in this case is:

```
compilation unit [COMPILE_UNIT]:38654705665 @0:0
   com.goldencode.p2j.util.* [KW_IMPORT]:38654705667 @0:0
   com.goldencode.p2j.util.BlockManager.* [STATIC_IMPORT]:38654705668 @0:0
   DynGenQuery [KW_CLASS]:38654705669 @0:0
       [CS_INSTANCE_VARS]:38654705673 @0:0
```

```
        = [ASSIGN]:38654705717 @0:0
           AdaptiveQuery [REFERENCE_DEF]:38654705718 @0:0
           AdaptiveQuery [CONSTRUCTOR]:38654705719 @0:0
      [CS_INSTANCE_METHODS]:38654705676 @0:0
        execute [METHOD_DEF]:38654705678 @0:0
            [BLOCK]:38654705679 @0:0
              initialize [METHOD_CALL]:38654705722 @0:0
                 query0 [REFERENCE]:38654705723 @0:0
                 tt [REFERENCE]:38654705724 @0:0
                 ? [STRING]:38654705725 @0:0
                  [NULL_LITERAL]:38654705726 @0:0
                 tt.recid asc [STRING]:38654705727 @0:0
                  [EXPRESSION]:38654705728 @0:0
                    new Object [REFERENCE_DEF]:38654705729 @0:0
                     [INITIALIZER]:38654705730 @0:0
                       P2JQuery.Parameter [CAST]:38654705731 @0:0
                           [LAMBDA]:38654705732 @0:0
                              [LPARENS]:38654705733 @0:0
                             isEqual [STATIC_METHOD_CALL]:38654705735 @0:0
                                logical [CONSTRUCTOR]:38654705736 @0:0
                                   ControlFlowOps.invokeDynamicFunctionIn [STATIC_METHOD_CALL]:38654705737 @0
:0
                                      hqlParam1dq [REFERENCE]:38654705738 @0:0
                                      thisProcedure [STATIC_METHOD_CALL]:38654705739 @0:0
                                      hqlParam2dq [REFERENCE]:38654705740 @0:0
                                hqlParam3dq [REFERENCE]:38654705741 @0:0
   com.goldencode.p2j.persist.* [KW_IMPORT]:38654705705 @0:0
   com.goldencode.testcases.dmo._temp.* [KW_IMPORT]:38654705706 @0:0
   com.goldencode.p2j.util.CompareOps.* [STATIC_IMPORT]:38654705742 @0:0
   com.goldencode.p2j.util.ProcedureManager.* [STATIC_IMPORT]:38654705743 @0:0
   com.goldencode.p2j.persist.lock.* [KW_IMPORT]:38654705744 @0:0
```

Note the use of invokeDynamicFunctionIn instead of invokeDynamicFunction.

Is it safe for me to modify the DYNAMIC_CALL condition to check for any of these variants? My inclination is yes, since it appears all the parameters in the interpreter are managed dynamically. But I wanted to get your advice before making this assumption and modifying the code.

#2 - 05/03/2021 07:16 PM - Ovidiu Maxiniuc

Eric Faulhaber wrote:

> Is it safe for me to modify the DYNAMIC_CALL condition to check for any of these variants? My inclination is yes, since it appears all the parameters in the interpreter are managed dynamically. But I wanted to get your advice before making this assumption and modifying the code.

Yes, it should be. I do not see any reason it would not work.
You may want to replace the if (DYNAMIC_CALL.equals(next.getText())) line with if (next.getText().startsWith(DYNAMIC_CALL)) in RuntimeJastInterpreter:332 and boolean dynamicCall = (dynamicCalls != null) && DYNAMIC_CALL.equals(methodName); line with boolean dynamicCall = (dynamicCalls != null) && methodName.startsWith(DYNAMIC_CALL); at RuntimeJastInterpreter:1141. And since DYNAMIC_CALL is now a prefix marker instead of the actual marker, we probably should rename it accordingly.

#3 - 05/04/2021 07:52 PM - Eric Faulhaber

- Assignee set to Eric Faulhaber

- Status changed from New to WIP

#4 - 05/04/2021 08:21 PM - Eric Faulhaber

- % Done changed from 0 to 100

- Status changed from WIP to Closed

Fixed in 3821c/12370. Confirmed the fix in a customer scenario.