Database - Bug #6492

problems with dynamic queries for meta tables

06/06/2022 10:30 AM - Constantin Asofiei

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

Description

History

#2 - 06/06/2022 10:39 AM - Constantin Asofiei

- Start date deleted (06/06/2022)

You can have multiple connected databases (like p2j_test and fwd), a loop which iterates through all these and:

- create a buffer for that database meta table, like create buffer hb for table "p2j_test._sequence".
- do a hb:find-first("where _seq-name = 'seq1'").

The dynamic conversion in FWD has two problems:

- when loading the .p2o for a file (via SchemaDictionar.loadFromAst), the temp-table will be added as _temp to SchemaDictionary.databases, instead of being prefixed with @_temp.
- the table lookup fails with an AmbiguousSchemaNameException (as it looks for unqualified _sequence table), and in SchemaDictionary.findEntry, there is this code:

```
         if (entityName.getDatabase() == null && name.startsWith("_"))
         {
            String qualified = null;
            String lastDatabase = getLastDatabase();
```

The problem is with getLastDatabase() - when converting such a query, we should have 'preferred database(s)' from the explicit buffers added to this query. But, getLastDatabase() finds the last entry in SchemaDictionar.databases - which, in some cases, is _temp.

I don't know if having _temp instead of @_temp is a real problem or not, as the root cause above is the fact that the SchemaDictionary is looking through all databases, not just the ones in the query's buffers.

Eric/Ovidiu: can you advise where I should look to make the query buffer databases 'preferred' at SchemaDictionary?

#5 - 04/25/2024 06:48 PM - Ovidiu Maxiniuc

This particular issue was fixed in 8437b/15175, by qualifying the table name with its database (except for temporary tables, where this is not possible).

However, in the general case, a query may have multiple buffers and their name cannot be qualified ... at least not without parsing the query first. If:

- a single table is used (the particular case of simple find);
- the full query is performed on a single persistent database (all subsequent buffers belong to same database);
- or the name collision / ambiguity can be caused only by the first buffer,

we can "hint" the resolution algorithm with a preferred database name like this:

```
=== modified file 'src/com/goldencode/p2j/persist/DynamicQueryHelper.java'
--- old/src/com/goldencode/p2j/persist/DynamicQueryHelper.java    2022-05-26 23:30:04 +0000
+++ new/src/com/goldencode/p2j/persist/DynamicQueryHelper.java    2022-08-15 19:45:35 +0000
@@ -323,6 +323,7 @@
          WorkArea      wa         = locate();
          long          queryId    = queryCounter.getAndIncrement();
          AstManager    astManager = AstManager.get();
+         SchemaDictionary    dict = null;

          // set the info about what we are currently parsing
          wa.predicate    = predicate;
@@ -338,7 +339,26 @@
          {
             StringBuilder pCode = new StringBuilder();
             SymbolResolver sym = SymbolResolver.newRuntimeInstance(); // build the symbol resolver
-            SchemaDictionary dict = RecordBuffer.getSchemaDictionary(allBuffers);
+            dict = RecordBuffer.getSchemaDictionary(allBuffers);
+
+            // attempt to detect the database we are working on:
+            String prefDb = null;
+            for (Buffer buffer : buffers)
+            {
+               if (prefDb == null)
+               {
+                  prefDb = ((BufferImpl) buffer).buffer().getDatabase().getName();
+               }
+               else if (!prefDb.equals(((BufferImpl) buffer).buffer().getDatabase().getName()))
+               {
+                  prefDb = null;
+                  break;
+               }
+            }
+            if (prefDb != null)
+            {
+               dict.setPreferredDatabase(prefDb);
+            }
             sym.setSchemaDictionary(dict); // set the schema dictionary used by this context

             try
@@ -535,6 +555,12 @@
             wa.jastFile = null;
             wa.astFile = null;
             wa.p2oFile = null;
+
+            if (dict != null)
+            {
+               // reset preferred database
+               dict.setPreferredDatabase(null);
+            }
          }
       }
       else

=== modified file 'src/com/goldencode/p2j/schema/SchemaDictionary.java'
--- old/src/com/goldencode/p2j/schema/SchemaDictionary.java    2022-07-29 10:49:09 +0000
+++ new/src/com/goldencode/p2j/schema/SchemaDictionary.java    2022-08-15 19:34:53 +0000
@@ -866,6 +866,9 @@
    /** Cache of temp-table added scopes. */
    private final Map<String, Scope> cachedScopes = new HashMap<>();

+   /** The prefered database. If {@code null} the last connected is assumed. */
+   private String prefDb = null;
```

```
+
    /**
     * Create a schema dictionary based upon a single database schema. This
     * is necessary during schema parsing when resolving inline Progress code
@@ -4598,7 +4601,7 @@
          if (entityName.getDatabase() == null && name.startsWith("_"))
          {
             String qualified = null;
-            String lastDatabase = getLastDatabase();
+            String lastDatabase = getPreferedDatabase();

             switch (type)
             {
@@ -4649,6 +4652,18 @@
     }

     /**
+
     * Configures the preferred database. If set to {@code null} the prefered database is computed dynamically,
+     * being the last connected database.
+     *
+     * @param   prefDb
+     *          The new preferred database or {@code null} to enter auto-detection mode.
+     */
+    public void setPreferredDatabase(String prefDb)
+    {
+       this.prefDb = prefDb;
+    }
+
+    /**
      * Get the database qualifier for the last "connected" (i.e. loaded/configured) database, which is not
      * a temp/work table database.
      * <p>
@@ -4659,8 +4674,13 @@
      *
      * @return  The database qualifier for the last connected database.
      */
-    private String getLastDatabase()
+    private String getPreferedDatabase()
     {
+       if (prefDb != null)
+       {
+          return prefDb;
+       }
+
        String lastDatabase = null;

        for (String next : databases.keySet())
```

This works well in the above cases, but other, more complex queries can be constructed, like this

```
create query qh2.
qh2:set-buffers(buffer db1.book:handle, buffer db2._file:handle).
qh2:query-prepare("for each book, each _file").
qh2:query-open()
```

The algorithm will choose as preferred database the database of first encountered buffer db1. The problem arise when the _file is incorrectly resolved from same database. This will not be noticeable because, in this case the table structure is identical, but the result of the query is likely incorrect. This kind of errors are difficult to notice and, therefore the code is dangerous to be kept.

Since the sample above is valid in 4GL, we need to find a solution for parsing these kind of queries.