Database - Bug #5121

Converted temp table should be using the legacy name instead of the converted name in displayed messages

01/29/2021 08:51 AM - Roger Borrello

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		

Description

History

#1 - 01/29/2021 08:52 AM - Roger Borrello

Saw this message logged: Unable to locate shared temp-table or work-table definition for table <converted tablename> in procedure <procedure name>. (1429).

#3 - 03/10/2021 12:02 PM - Roger Borrello

At this point in processing I don't think we know the legacy name (since we haven't found it).

From SharedVariableManager:

```
   public static <T extends Temporary> T lookupTempTable(String name, Class<T> expected)
   throws ErrorConditionException
   {
      Runnable error = () ->
      {
         String pname = ProcedureManager.getInstantiatingExternalProgram();
         String msg = "Unable to locate shared temp-table or work-table definition " +
                      "for table %s in procedure %s";
         msg = String.format(msg, name, pname);

         errorHelper(1429, msg);
      };

      return lookupTempTable(name, expected, error);
   }
```

#4 - 03/10/2021 01:17 PM - Greg Shah

I think the legacy name can be found in expected. It is the DMO we are searching for, which inherently maps 1-to-1 with the legacy table.

#5 - 03/10/2021 03:51 PM - Roger Borrello

Greg Shah wrote:

> I think the legacy name can be found in expected. It is the DMO we are searching for, which inherently maps 1-to-1 with the legacy table.

I looked throughout expected for something that looked like it would show the legacy information.

#6 - 03/10/2021 04:08 PM - Greg Shah

Eric?

#7 - 03/10/2021 09:22 PM - Eric Faulhaber

Roger Borrello wrote:

> Greg Shah wrote:
>
> > I think the legacy name can be found in expected. It is the DMO we are searching for, which inherently maps 1-to-1 with the legacy table.
>
> I looked throughout expected for something that looked like it would show the legacy information.

expected is going to be a grouping interface that combines the main DMO interface (which defines the getters/setters and has all the annotations), and the TempTableBuffer interface. TempTableBuffer is itself is a grouping interface that combines the Temporary marker interface with the Buffer interface (which defines a lot of worker methods common to buffers). The DMO and Buffer interfaces are grouped this way so that we can invoke Buffer methods and DMO getter/setter methods on the same proxy object from business logic, but the DMO implementation class doesn't need to define the Buffer methods.

Anyway...the legacy table name is the legacy member of the @Table annotation on the DMO interface. Based on the interface hierarchy described above, it is a little painful and not very efficient to get at, so we look it up once when the DMO interface is loaded and the DMO implementation class is assembled for it. It is cached in a DmoMeta object, so that later lookups can be very fast.

However, in this case, expected is passed to SharedVariableManager.lookupTempTable from TemporaryBuffer.useShared *before* the associated record buffer is instantiated, and thus before the DmoMeta object is associated with it. If we had that RecordBuffer instance, it would be a simple matter to access it with recordBuffer.getDmoInfo().legacyTable.

Ovidiu, can you think of an efficient way to give access to SharedVariableManager.lookupTempTable to the correct DmoMeta instance at this early stage?

#8 - 03/11/2021 07:49 AM - Ovidiu Maxiniuc

Eric Faulhaber wrote:

> Ovidiu, can you think of an efficient way to give access to SharedVariableManager.lookupTempTable to the correct DmoMeta instance at this early stage?

With local data, the only way to retrieve the ABL table name is:

```
DmoMetadataManager.getDmoInfo(expected).legacyTable
```

It is not a direct access as it uses a map lookup, but this it only used in exceptional cases (in case of an error) so I think this is acceptable.

Of course, if we could enrich the lookupTempTable(String name, Class<T> expected) signature to accommodate the RecordBuffer instance or its DmoMeta member to access it directly, but I do not think this worth the trouble of carrying extra parameter along since it will be used only in exceptional cases. However, since the instance is not yet fully constructed this is a dead end.