Conversion Tools - Bug #5551

Case sensitivity of default buffer name

07/16/2021 09:52 AM - Ovidiu Maxiniuc

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

#1 - 07/16/2021 10:03 AM - Ovidiu Maxiniuc

A couple of weeks ago, while running some tests and I notice minor differences in the output between 4GL and FWD. These are the name of the default buffers for permanent tables. For example, for

```
   ADD TABLE "Book"
```

definition in my .df file I get the following definition in generated Java code:

```
   Book.Buf book = RecordBuffer.define(/*dmoBufIface=*/Book.Buf.class, /*database=*/"fwd", /*variable=*/"book"
, /*legacyName=*/"book");
```

Strangely, this happens only for the default buffer. For all other buffers, the legacy name is kept. Including the temporary buffers, although I cannot be sure for the dynamically created tables which do not have an explicit define Java statement. For example, for temp-table buffers like:

```
   DEFINE BUFFER bAak FOR book.
```

the generated java code is

```
   Book.Buf baak = RecordBuffer.define(/*dmoBufIface=*/Book.Buf.class, /*database=*/"fwd", /*variable=*/"baak"
, /*legacyName=*/"bAak");
```

My question was: is there a reason for normalizing it? I wished to investigate this and fix it, if possible.

#2 - 07/16/2021 10:11 AM - Ovidiu Maxiniuc

After preliminary investigations, I traced that the legacy buffer name (legacybufname) is computed in convert/buffer_definitions.rules:620 by dropping the schema name from the bufname annotation. At this moment the buffer name is already lowercased.

Stepping backward at the moment the annotation is set. In progress.g:26829:

```
    String bufname  = sym.lookupBufferName(tablename, false, forceTemp, forcePersistent);
```

The call stack is like this:

```
      at com.goldencode.p2j.schema.NameNode.getName(NameNode.java:244)
      at com.goldencode.p2j.schema.SchemaDictionary.getRecordName(SchemaDictionary.java:4169)
      at com.goldencode.p2j.schema.SchemaDictionary.getBufferForTable(SchemaDictionary.java:3192)
      at com.goldencode.p2j.uast.SymbolResolver.lambda$6(SymbolResolver.java:5980)
      at com.goldencode.p2j.uast.SymbolResolver$$Lambda$32.117249632.process(Unknown Source:-1)
      at com.goldencode.p2j.uast.SymbolResolver.processHierarchy(SymbolResolver.java:9229)
      at com.goldencode.p2j.uast.SymbolResolver.lookupBufferName(SymbolResolver.java:5985)
```

If we look at the method at the top, it reads:

```
    return ast.getText().toLowerCase();
```

Also, the method javadoc expressly mentions *'always lower case'*. However, the text of the ast node is already in lowercase (at least for my case):

```
    book [TABLE]:8589936069 @963:5
          (legacy_name=Book, historical=Book)
          [...]
```

Notice the annotations. Both of them contain the information we need, but it is not accessible with current API.

I thought replacing

```
    buf.append(table.getName());
```

in SchemaDictionary.getRecordName(NameNode node, int type, String name) (line 4169) with

```
    buf.append(table.getAst().getAnnotation("historical"));
```

would fix the issue. The problem is, at the same time, other changes in generated code happen: I noticed some variables has changed their casing as well. For example, a legacy buffer named sT-book (I use this often this kind of names as it has both different casing and - in its name) will be converted to sTBook with the patch instead of stBook with FWD from repository. Since all occurrences of the affected buffer are updated, the code seems fine and compiles without issues.

#3 - 07/16/2021 10:43 AM - Ovidiu Maxiniuc

At this moment, my isolated testcases passed. A problem was detected in a customer project with the dynamic temp-tables: they do not have either legacy_name or historical annotations.
I added it in DynamicConversionHelper:943:

```
            table.putAnnotation("historical", legacyTableName);
```

and use the unaltered legacy name for bufname annotation in defChild constructed in prepareTree(). Notice here,

```
         String legacyTableName = TextOps.rightTrimLower(TableMapper.getLegacyName(rbuf));
```

variable will NOT preserve the casing of the name.

Also, in same method there is the following construct:

```
            if (legacyBufName.equalsIgnoreCase(legacyTableName))
            {
               defChild.putAnnotation("force_dmo_alias", rbuf.getDMOAlias());
            }
```

It really does not make sense to force an alias if the buffer and table name are matching so it seems like the condition is reversed. Beside that, this code is used with already existing buffers and the method only created the tree to be used in a dynamic conversion. The 4GL buffer names are the same so the target java name of the buffer must also be a match for the name of the buffer sent as parameter. If this reside in a sensible scope, it is possible that the parameter buffer to have a decorated name (like book__1) so enforcing the known alias (variable name) is pretty mandatory at this moment.

Although this seemed enough, the dynamic runtime still failed. Stack traces like:

```
com.goldencode.p2j.persist.RuntimeJastInterpreter$InterpreterException: No REFERENCE named (tmpbookbuf2)
        at com.goldencode.p2j.persist.RuntimeJastInterpreter.evalExpression(RuntimeJastInterpreter.java:1439)
        at com.goldencode.p2j.persist.RuntimeJastInterpreter.collectParameters(RuntimeJastInterpreter.java:850
)
        at com.goldencode.p2j.persist.RuntimeJastInterpreter.callMethod(RuntimeJastInterpreter.java:941)
        at com.goldencode.p2j.persist.RuntimeJastInterpreter.execMethod(RuntimeJastInterpreter.java:658)
        at com.goldencode.p2j.persist.RuntimeJastInterpreter.interpret(RuntimeJastInterpreter.java:418)
        at com.goldencode.p2j.persist.DynamicQueryHelper.lambda$parse$5(DynamicQueryHelper.java:554)
        at com.goldencode.p2j.persist.AbstractQuery.notifyQueryOpenListeners(AbstractQuery.java:864)
        at com.goldencode.p2j.persist.QueryWrapper.queryOpen(QueryWrapper.java:4237)
        [...]
```

were reported in #5034-904, #4319-16. The actual buffer was named tmpBookBuf2 but, apparently, the dynamic conversion failed to 'force' the buffer name as instructed by force_dmo_alias annotation (note that the annotation is visible in the tree dump of the AST used by the RuntimeJastInterpreter).

Because the code changes were blocking the main branch, these changes were rolled back and this tracker added.