Base Language - Feature #4378

properly handle clob/lonchar assignment, especially the implicit codepage conversion

11/02/2019 01:06 PM - Greg Shah

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

Description

Related issues:

Related to Base Language - Feature #3753: I18N additions	**Closed**
Related to Database - Feature #3958: BUFFER-COPY optimization	**Closed**

History

#1 - 11/02/2019 01:06 PM - Greg Shah

- Related to Feature #3753: I18N additions added

#2 - 11/02/2019 01:07 PM - Greg Shah

- When a longchar/clob value is assigned to a clob field, it is implicitly converted into the codepage of the target field. I suppose we'll have to handle it in RecordBuffer.invoke in the future.
- We will need to handle this in assignments in both directions (e.g. also TO longchar) if we don't already do it properly. This means longchar.assign() needs to be modified.
- FWD's implementation of BUFFER-COPY uses the RecordBuffer invocation handler for its individual fields, so if we implement it in the invocation handler, we are covered for BUFFER-COPY.

The original discussion was from #3753-108.

#3 - 07/15/2020 12:46 PM - Greg Shah

- Related to Feature #3958: BUFFER-COPY optimization added

#4 - 07/15/2020 12:46 PM - Greg Shah

> FWD's implementation of BUFFER-COPY uses the RecordBuffer invocation handler for its individual fields, so if we implement it in the invocation handler, we are covered for BUFFER-COPY.

Is this still true in a post-4011 world?

#5 - 11/07/2020 08:56 AM - Greg Shah

- Assignee set to Ovidiu Maxiniuc

#6 - 02/02/2021 04:47 PM - Ovidiu Maxiniuc

I encountered a conversion issue which I think is really related to this tracker.

If we have a fixedLongchar variable and a handle to a buffer with a longCharFiled field of type longchar the 2nd line of following code will fail to convert properly:

```
FIX-CODEPAGE(fixedLongchar) = 'utf-8' NO-ERROR.
FIX-CODEPAGE(hHandle::longCharFiled) = 'utf-8' NO-ERROR.
```

The resulting code is:

```
silent(() -> fixedLongchar.fixCodePage("utf-8"));
silent(() -> hHandle.unwrapDereferenceable().dereference("longCharFiled", new
 character(hHandle.unwrapDereferenceable().dereference("longCharFiled")).fixCodePage("utf-8")));
```

Evidently, the second line is really broken. It should be read something like:

```
silent(() -> hHandle.unwrapDereferenceable().dereference("longCharFiled").fixCodePage("utf-8")));
```

but there is a problem: the result of dereference() is BaseDataType so fixCodePage() is not accessible.
Probably we need to implement this method as a static method instead:

```
silent(() -> longchar.fixCodePage(hHandle.unwrapDereferenceable().dereference("longCharFiled"), "utf-8"));
```

where fixCodePage will take a BaseDataType as its first argument and cast it internally to longchar, eventually raising the appropriate error conditions if the type is incorrect.

At any rate, this construct shows that other similar method can fail to compile if a specific member method of a BDT is attempted to be called when this value is the result of dereferenciation.

#7 - 02/02/2021 04:58 PM - Constantin Asofiei

Can I assume that FIX-CODEPAGE(hHandle::longCharFiled) = 'utf-8' NO-ERROR. will alter the codepage of longCharFiled? If so, the static longchar.fixCodePage method will not work, as the dereference will return a copy of the value.

#8 - 02/02/2021 05:59 PM - Ovidiu Maxiniuc

I guess so. I have not yet investigated how the FIX-CODEPAGE function works, just spotted the compile-time error and tried to find a solution. But you are probably right. This means the current code does not work since the getters will always return copies of the values of the fields and variables.

The static method would fix the case when the variables and fields are set by reference (by invoking internal API unavailable to 4GL programmer), but I do not know if we can handle the dereference case.

#9 - 02/25/2021 07:51 PM - Ovidiu Maxiniuc

Actually the dereference operation pose other issues. The COPY-LOB is also not working correctly when a dereferenced parameter is used. A construct like:

```
   COPY-LOB FROM FILE 'resources/file.dat' TO ttBuffer::blob-field.
```

will be converted to:

```
   new LobCopy(new SourceLobFile("resources/file.dat"), new
 TargetLob(ttBuffer.unwrapDereferenceable().dereference("blob-field"))).run();
```

Because the dereference will return a (temporary) **copy of** the blob field to LobCopy constructor as a BDT object, and only that will be set up; the record stored in ttBuffer will not be touched, that is, its field will remain unchanged.

#10 - 04/07/2021 10:52 AM - Greg Shah

Please update the % Done.

#11 - 04/08/2021 12:06 AM - Ovidiu Maxiniuc

- Status changed from New to WIP

- % Done changed from 0 to 100

The COPY-LOB issues (conversion and dereferenciation) are fixed in a previous commit 3821c/12222.

Instead of changing longchar.assign() the methods had to be implemented in clob to be aware of the codepage.

I finished testing and cleaning up yesterday a patch which makes sure the CLOB fields are initialized with the proper CODEPAGE before any text is set, when they are returned as copies of buffer data.

All later changes can be found in 3821c, revision 12257.

#12 - 04/08/2021 06:52 AM - Greg Shah

- Status changed from WIP to Closed