Database - Bug #5891

denormalized properties of extent field cause NPE when accessing attributes of buffer fields

12/13/2021 05:14 PM - 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

#2 - 12/13/2021 05:28 PM - Ovidiu Maxiniuc

While testing the small fix for #5890 I encountered the issue described below. It manifested only in my testcase - not with customer code, but I may surface in other places.

The issue occurs when attempting to access extent buffer field attributes of dynamically constructed temp tables. It starts when the DmoMeta stores its fields (or properties, in fact) in **denormalized** format. Hence, an extent field f1 extent 3 will be represented as 3 different properties (like field21, field23, and field23). From here it is used by TableMapper (see loadFields(DmoMeta dmoInfo)) to construct its internal data structure, property-indexed.

The problem manifests only when the buffer-field (bufferField()) method is invoked. It uses indexed or legacy-name index to locate the fields. At any rate, when the BufferFieldImpl() constructor is called, it will initialize the lfi member using computeLegacyInfo(), using the property name of the original property (field2). But this does not exist because it was denormalized, so lfi remains null. This means the whole BufferFieldImpl is broken, all accesses to the attributes (which are stored in lfi) will cause a NPE to be thrown.

In other words, we loose the link to original field and its attributes when the extent property is denormalized. I identified this using dynamic temp-tables, but the issue is probably present for all denormalized extent fields. I am trying to think of a solution here.

There might be another, secondary issue here, related to the way the denormalized fields get their property names. We need lots of fields. The first declared is an extent 2, so two denormalized properties will be created: field11 and field12. Then, if there are another 11 plain fields. Their property names will be field2, field3, ... field10, field11, and field12. As you can see, we have a naming collision here (field11 and field12 map both the denormalized properties of the field1 and the plain field11 and field12 properties). The solution to fix this is probably as simple as adding an underscore before the index of the denormalized properties so that they become field1_1 and field1_2.

#3 - 12/14/2021 02:08 PM - Ovidiu Maxiniuc

Here is a testcase which will fail with NPE, as described above:

```
define variable tth as handle.
define variable dbh as handle.
```

```
define variable dbh-1 as handle.

create temp-table tth.
tth:add-new-field("my-fielD", "character", 2).
tth:TEMP-TABLE-PREPARE("customM-temP-tablE").
dbh = tth:DEFAULT-BUFFER-HANDLE.
dbh-1 = dbh:buffer-field(1).

message dbh-1:NAME
        dbh-1:DATA-TYPE
        dbh-1:EXTENT
        dbh-1:DEFAULT-STRING.
```

My second concern from note-2 is not real. The name of the property field<N> is computed during the dynamic conversion and the names are unique. In the event of a property name collision the latter property names receives a suffix.