Database - Bug #11745

two temp-tables with the same structure but different NO-UNDO emit NO-UNDO sql (undoable state is lost)

08/13/2026 06:31 PM - Constantin Asofiei

Status:	Test	**Start date:**	
Priority:	High	**Due date:**	
Assignee:	Teodor Gorghe	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**reviewer:**	Constantin Asofiei
vendor_id:	GCD	**production:**	No
case_num:		**env_name:**	
version_reported:		**topics:**	
version_resolved:			

Description

Related issues:

Related to Database - Bug #8593: H2 performance improvement for table 'trunca...	**Test**
Related to Base Language - Bug #10935: Inherit the NO-UNDO option for a temp-...	**Closed**

History

#1 - 08/13/2026 06:31 PM - Constantin Asofiei

- Related to Bug #8593: H2 performance improvement for table 'truncate by multiplex' added

#4 - 08/13/2026 06:36 PM - Constantin Asofiei

Convert these two in different files:

1. def temp-table ttbb field f1 as int.
2. def temp-table ttbb no-undo field f1 as int.

They will have both the same DMO, with noUndo = true annotation. The SQL will be created from the DMO, and not from the TemporaryBuffer.define - thus both will be no-undo!

This patch tries to solve it:

```
Index: rules/include/common-progress.rules
===================================================================
--- rules/include/common-progress.rules    (revision 4897)
+++ rules/include/common-progress.rules    (working copy)
@@ -12023,7 +12023,8 @@
                                       'xml-node-name',
                                       'serialize-name',
                                       'before-table',
-                                      'after-table')
+                                      'after-table',
+                                      'no-undo')
          </rule>
          <rule>tmpTabCrit.put(prog.temp_table, tmpTabProps)</rule>
          <rule>tmpTabCrit.put(prog.work_table, tmpTabProps)</rule>
Index: rules/schema/fixups.xml
===================================================================
--- rules/schema/fixups.xml    (revision 4897)
+++ rules/schema/fixups.xml    (working copy)
@@ -798,7 +798,7 @@
             <action>execLib('annotate-boolean', 'serialize-hidden', 1)</action>
          </rule>
```

```
-          <!-- convert serialize-hidden AST to an annotation -->
+          <!-- convert serialize-name AST to an annotation -->
           <rule>type == prog.kw_serialzn
              <rule>parent.type == prog.properties
                 <action on="true">level = 2</action>
@@ -806,6 +806,16 @@
              </rule>
              <action>execLib('annotate-string', 'serialize-name', null, level)</action>
           </rule>
+
+          <!-- convert no-undo AST to an annotation -->
+          <rule>type == prog.kw_no_undo
+             <rule>parent.type == prog.properties
+                <action on="true">level = 2</action>
+                <action on="false">level = 1</action>
+             </rule>
+             <!-- leave it at the PROPERTIES, as that is what it is used to emit the NO-UNDO at the DMO -->
+             <action>copy.getAncestor(level).putAnnotation('no-undo', true)</action>
+          </rule>

           <!-- convert xml-data-type AST to an annotation -->
           <rule>type == prog.kw_xml_dtyp
```

But it contradicts this commit:

```
revno: 15178 [merge]
committer: Constantin Asofiei <ca@goldencode.com>
branch nick: trunk
timestamp: Mon 2024-04-29 19:30:03 +0300
message:
  Reverted the changes in trunk rev 15170 related to temp-table NO-UNDO emitted at DMO annotation - these must
 be emitted at the master buffer. Refs #8593 #8363 #8700
```

So it needs more investigations. I think we need to remove the DMO annotation and use only and only the TemporaryBuffer.define flag, always.

#6 - 08/14/2026 02:54 AM - Constantin Asofiei

For the structure SQL (create DDL) we can have two sets: one with no-undo and one without. And choose depending if the DMO instance is defined undoable or not.

#7 - 08/14/2026 02:08 PM - Constantin Asofiei

- Assignee set to Teodor Gorghe

- Priority changed from Normal to High

Teodor, please work on this. Lets get rid of the no-undo annotation and do everything at runtime based on the flag at TemporaryBuffer.define.

#8 - 08/17/2026 01:17 AM - Teodor Gorghe

- Related to Bug #10935: Inherit the NO-UNDO option for a temp-table defined using the LIKE temp-table option added

#9 - 08/17/2026 02:37 AM - Teodor Gorghe

Constantin, since we will have two temp-tables, with different SQL schema definition (+ transactional at the end), they will resolve to the same SQL table name. We need to add something to the SQL name, to avoid collision, like __nu, or a signature from the procedure name, right?

#10 - 08/17/2026 02:49 AM - Constantin Asofiei

Teodor Gorghe wrote:

> Constantin, since we will have two temp-tables, with different SQL schema definition (+ transactional at the end), they will resolve to the same SQL table name. We need to add something to the SQL name, to avoid collision, like __nu, or a signature from the procedure name, right?

Yes, we will need to have different temp-tables in H2. Lets keep the DMOs unchanged and add the suffix at runtime (so if NO-UNDO, you will have __nu suffix in the SQL name and also the NO-UNDO option at the CREATE TABLE).

#11 - 08/17/2026 08:35 AM - Teodor Gorghe

The main problem when following this approach is that 4GL allows changing the undoable state by changing the value of UNDO attribute. If we use two distinct SQL tables, it will involve moving the records from one table to another one (at least).

Also, lots of files from persist layer needs to be changed to append __nu suffix.

#12 - 08/17/2026 08:38 AM - Constantin Asofiei

Teodor Gorghe wrote:

> The main problem when following this approach is that 4GL allows changing the undoable state by changing the value of UNDO attribute. If we use two distinct SQL tables, it will involve moving the records from one table to another one (at least).

You mean after you have created and prepared the dynamic temp-table? Or even for a static temp-table?

> Also, lots of files from persist layer needs to be changed to append __nu suffix.

Is this because of the dependence on the DMO SQL name annotation? IMO we should get rid of that, too, and let it be calculated at runtime.

#13 - 08/17/2026 08:40 AM - Teodor Gorghe

Constantin Asofiei wrote:

> You mean after you have created and prepared the dynamic temp-table? Or even for a static temp-table?

I mean for dynamic temp-tables. The alternative for that is a H2 change which marks the multiplex as undoable / no-undo.

> Is this because of the dependence on the DMO SQL name annotation? IMO we should get rid of that, too, and let it be calculated at runtime.

Yes.

#14 - 08/17/2026 08:45 AM - Constantin Asofiei

Argh, Alex, UNDO option is modifiable for the lifetime of a temp-table!

```
def temp-table tt1 field f1 as int.

create tt1.
tt1.f1 = 10.
release tt1.

def var htt as handle.
create temp-table htt.
htt:create-like("tt1").
htt:temp-table-prepare("rtt1").
htt:undo = true.
message  htt:undo.

message temp-table tt1:undo. // yes

temp-table tt1:undo = false.

message temp-table tt1:undo. // no
```

#15 - 08/17/2026 08:46 AM - Constantin Asofiei

Teodor: go ahead and fix this as needed. So, DMO annotation for undoable is still not needed. And the undoable state must be tracked, live, at the multiplex ID in H2, and not at the CREATE TABLE DDL. Correct?

#16 - 08/17/2026 08:47 AM - Teodor Gorghe

Right, that is what I am thinking.

#17 - 08/17/2026 08:55 AM - Alexandru Lungu

Constantin Asofiei wrote:

> Argh, Alex, UNDO option is modifiable for the lifetime of a temp-table!
> [...]

Apparently documentation states this. But it can be changed outside a transaction, **or within a transaction** if the table is empty.

#18 - 08/18/2026 03:58 AM - Constantin Asofiei

Alexandru Lungu wrote:

> Constantin Asofiei wrote:
>
> > Argh, Alex, UNDO option is modifiable for the lifetime of a temp-table!
> > [...]
>
> Apparently documentation states this. But it can be changed outside a transaction, **or within a transaction** if the table is empty.

Looks like this is already implemented in AbstractTempTable.setCanUndo.

#19 - 08/18/2026 03:59 AM - Teodor Gorghe

- File fwd-h2-11745.patch added

Proposed H2 changes. I will attach the FWD changes later on.

#20 - 08/18/2026 04:02 AM - Constantin Asofiei

Please create a H2 branch as ~/secure/code/p2j_repo/fwd-h2/active/11745a_h2/

#21 - 08/18/2026 06:27 AM - Teodor Gorghe

Created task branch **11745a_h2**, based of **p2j_repo/fwd-h2/trunk** rev **64** and committed revision **65**:

- Make NO-UNDO flag to be multiplex-wise and updatable.

#22 - 08/18/2026 06:48 AM - Teodor Gorghe

- Status changed from New to WIP

- % Done changed from 0 to 100

- reviewer Constantin Asofiei added

Created task branch **11745a** and committed revision **16704**:

- Removed undoable from dmo table annotation and now the flag relies on the TemporaryBuffer.define argument and the UNDO attribute.

Constantin, can you review the H2 changes and the current changes from **11745a**?
Note that you would need to patch the build.gradle to resolve to the new H2 build, since we didn't published the H2 artifact.

Also, there is a preexisting bug in trunk, regarding shared temp-table lookup, which requires changes in the conversion and runtime to add a new TemporaryBuffer.useShared(String, Class, boolean) overload. Should I proceed fixing that?

#23 - 08/18/2026 06:48 AM - Teodor Gorghe

- Status changed from WIP to Review

#24 - 08/18/2026 06:50 AM - Constantin Asofiei

Teodor Gorghe wrote:

> Also, there is a preexisting bug in trunk, regarding shared temp-table lookup, which requires changes in the conversion and runtime to add a new TemporaryBuffer.useShared(String, Class, boolean) overload. Should I proceed fixing that?

Please post a test for this.

Also, make sure the tests are converted to ABLUnit and committed to tests/table/ in the testcases project.

#25 - 08/18/2026 07:00 AM - Constantin Asofiei

Please post some details why you've decided to still keep the table-level NO-UNDO in H2.

#26 - 08/19/2026 01:29 AM - Teodor Gorghe

Constantin Asofiei wrote:

> Please post some details why you've decided to still keep the table-level NO-UNDO in H2.

I don't have an reason why I have kept table level H2. Should I remove it and rely only by multiplex UNDO?

#27 - 08/19/2026 03:46 AM - Constantin Asofiei

Teodor Gorghe wrote:

> Constantin Asofiei wrote:
>
> > Please post some details why you've decided to still keep the table-level NO-UNDO in H2.
>
> I don't have an reason why I have kept table level H2. Should I remove it and rely only by multiplex UNDO?

Yes. We don't want two H2 tables for the same DMO. And it will make things clearer.

Alex - should we leave in H2 the NO-UNDO option at the table, and we'll just not use it from FWD? It will add some small complexity in H2, but I don't think is different from what Teodor has already done in 11745a_h2 branch.

#28 - 08/19/2026 03:51 AM - Teodor Gorghe

I have found a regression of **11745a** which I am fixing right now. Also, I am adding more tests regarding UNDO flip and per multiplex undo in tests/persistence/temp_table/undo_state/.

#29 - 08/19/2026 05:36 AM - Teodor Gorghe

Strange error found during testing:

```
   |  ├─ deleteThenUndoLeaveKeepsDeletion  ** Requested to persist DMO of type com.goldencode.testcases.dmo._
temp.Ttmut_1_1__Impl__ with recid 6656. A different DMO instance of type com.goldencode.testcases.dmo._temp.Tt
mut_1_1__Impl__ is already bound to this session with the same recid
```

With the testsuite which I have make, there are 22 tests which are failing (from 102 tests), but with **11745a**, which is not committed yet, the number of failing tests has decreased just to 2. The remaining failures are related to error handling, we doing raise an error when we need to.

I have already removed the table-wise undoable flag on H2, but I am waiting for Alex's answer.

#30 - 08/19/2026 05:41 AM - Constantin Asofiei

Do you have trunk 16704 for #11632? There we tried to solve a similar problem.

If you have rev 16704, try disabling session reclaiming. We need to figure out the root cause.

#31 - 08/19/2026 05:48 AM - Teodor Gorghe

Committed the tests on **xfer testcases** as rev **1872**:

- Added persistence/temp_table/undo_state tests.

I ran with trunk rev 16706, with the session reclaiming disabled, but the issue still occurs.
Failing test is TestNoUndoTempTableMutations:deleteThenUndoLeaveKeepsDeletion.

#32 - 08/19/2026 06:16 AM - Alexandru Lungu

> Alex - should we leave in H2 the NO-UNDO option at the table, and we'll just not use it from FWD? It will add some small complexity in H2, but I don't think is different from what Teodor has already done in 11745a_h2 branch.

It is fine by me. NO-UNDO is a syntax that can be avoided, that is all. There is a very small parsing overhead ... very small.

#33 - 08/19/2026 06:17 AM - Constantin Asofiei

Alexandru Lungu wrote:

> Alex - should we leave in H2 the NO-UNDO option at the table, and we'll just not use it from FWD? It will add some small complexity in H2, but I don't think is different from what Teodor has already done in 11745a_h2 branch.

> It is fine by me. NO-UNDO is a syntax that can be avoided, that is all. There is a very small parsing overhead ... very small.

Actually, after discussing with Teodor, I think we need to remove it - it just adds another layer of complexity, as in H2 we will need to consider the table-level NO-UNDO.

#34 - 08/19/2026 06:22 AM - Alexandru Lungu

> | ├── deleteThenUndoLeaveKeepsDeletion ** Requested to persist DMO of type com.goldencode.testcases.dmo._temp.Ttmut_1_1__Impl__ with recid 6656. A different DMO instance of type com.goldencode.testcases.dmo._temp.Ttmut_1_1__Impl__ is already bound to this session with the same recid

If this is reproducible with trunk, please let me know. We see such issue at some customers and we are not able to reproduce. If you have a test that reproduces, it is a gold-mine.

#35 - 08/19/2026 06:22 AM - Teodor Gorghe

Yes, it is very reproduceable with the latest trunk and the **11745a** state which I currently have, fixes that issue.

#37 - 08/19/2026 06:27 AM - Alexandru Lungu

> Yes, it is very reproduceable with the latest trunk and the 11745a state which I currently have, fixes that issue.

Perfect! I will watch this task if you need further help from me. I think we shall have 11745a delivered asap to customers; let me know how to help fast track it.

#38 - 08/19/2026 06:29 AM - Alexandru Lungu

> Do you have trunk 16704 for #11632? There we tried to solve a similar problem.

Constantin, AFAIK, the customer in question has force-no-undo-temp-tables flag in directory.xml, which make all tables no-undo. Do this also happen in their docker deployment?

#39 - 08/19/2026 06:34 AM - Constantin Asofiei

Alexandru Lungu wrote:

> Constantin, AFAIK, the customer in question has force-no-undo-temp-tables flag in directory.xml, which make all tables no-undo. Do this also

> happen in their docker deployment?

Yes, both projects have this in directory.xml

#40 - 08/19/2026 06:36 AM - Alexandru Lungu

Then this task won't help any of that projects, right? Unless 11745a has other changes that affect reclaiming for no-undo.

#41 - 08/19/2026 06:47 AM - Teodor Gorghe

That test still fails with trunk, because the expected result is different, because the temp-table is undoable, but it doesn't fail with the session recid conflict error.

#43 - 08/19/2026 07:02 AM - Constantin Asofiei

Teodor Gorghe wrote:

> That test still fails with trunk, because the expected result is different, because the temp-table is undoable, but it doesn't fail with the session recid conflict error.

Please find a detailed explanation for the Requested to persist DMO of type. At the least, we will understand why it happened and keep it in mind if we see it in other places.

#44 - 08/19/2026 07:13 AM - Alexandru Lungu

Constantin Asofiei wrote:

> Teodor Gorghe wrote:
>
> > That test still fails with trunk, because the expected result is different, because the temp-table is undoable, but it doesn't fail with the session recid conflict error.
>
> Please find a detailed explanation for the Requested to persist DMO of type. At the least, we will understand why it happened and keep it in mind if we see it in other places.

COMPLETELY AGREED!

#45 - 08/19/2026 09:24 AM - Teodor Gorghe

The session cache error happens in the following situation (only on trunk):

- The DMO interface is declared to be undoable, but the definition of temp-table is no-undo (I have another testcase which uses the same DMO, but it is really undoable).
- According to the trunk state, the undoable flag from DMO is used to create the SQL, so H2 thinks is an undoable table
- FWD server-side uses UndoableProvider, which is set from the TemporaryBuffer.define, which in this case, is no-undo.
- By having this state mismatch, the second test method fails because it is expected to keep the records in the table (trunk rollbacks the records when it is not supposed to be).
- 4GL reclaims PKs. Since H2 thinks it is an undoable table, on the next create, it will assign the PK to the first PK from the batch because the whole batch was reclaimed.
- Session cache old record wasn't evicted yet because you know, FWD knows this is a no-undo table, which is an optimization to don't evict records which is supposed to be in the temp database. Reaches to that error when associating to the session.

#46 - 08/19/2026 09:53 AM - Constantin Asofiei

In one of the projects, there is a setCanUndo(true) call, even if it has force-no-undo-temp-tables; but this should be caught by TemporaryBuffer.isUndoable.

#47 - 08/19/2026 09:59 AM - Constantin Asofiei

Although I think in trunk we may have an existing problem in DynamicTablesHelper.generateSchemaAst - it uses the table's no-undo status, and not via TemporaryBuffer.

```
      if (!builder.isUndoable())
      {
         // TODO: if other properties must be added, extract this node outside if
         ProgressAst propsAst = new ProgressAst(new CommonToken(ProgressParserTokenTypes.PROPERTIES,
                                                                "properties"));
         tableSchema.addChild(propsAst);

         propsAst.addChild(new ProgressAst(new CommonToken(ProgressParserTokenTypes.KW_NO_UNDO, "no-undo")));
      }
```

This should no longer be a problem in 11745a, right?

#48 - 08/19/2026 10:00 AM - Teodor Gorghe

I have removed that section in **11745a**.

#49 - 08/20/2026 05:06 AM - Teodor Gorghe

Constantin, I have committed **11745a/16705** and **11745a_h2/67** (they match).

Please take a look.
I will also commit the remaining tests. I have found another issue in FWD, but it is unrelated to 11745a. I am committing the new tests and investigate the issue. It is related to the fact that a dynamic buffer for a temp-table defined as no-undo, but behind, is treated as undoable.

#51 - 08/20/2026 05:24 AM - Teodor Gorghe

Added more tests in testcases about temp-table undo: logslogs

```
added tests/persistence/temp_table/undo_state
added tests/persistence/temp_table/undo_state/MixedFlavorsNoUndo.p

added tests/persistence/temp_table/undo_state/MixedFlavorsUndoable.p
added tests/persistence/temp_table/undo_state/NoUndoRowsAfterUndo.p
added tests/persistence/temp_table/undo_state/NoUndoScopeReopen.p
added tests/persistence/temp_table/undo_state/SharedUndoCalleeCount.p
added tests/persistence/temp_table/undo_state/SharedUndoCalleeFields.p
added tests/persistence/temp_table/undo_state/SharedUndoCalleeN.p
added tests/persistence/temp_table/undo_state/SharedUndoCalleeU.p
added tests/persistence/temp_table/undo_state/SharedUndoCallerN.p
added tests/persistence/temp_table/undo_state/SharedUndoCallerRows.p
added tests/persistence/temp_table/undo_state/SharedUndoCallerU.p
added tests/persistence/temp_table/undo_state/TestDynamicTempTableUndoAttribute.cls
added tests/persistence/temp_table/undo_state/TestDynamicTempTableUndoDefault.cls
added tests/persistence/temp_table/undo_state/TestDynamicTempTableUndoFlip.cls
added tests/persistence/temp_table/undo_state/TestMixedUndoFlavorsShareOneDmo.cls
added tests/persistence/temp_table/undo_state/TestNoUndoSurvivesRepeatedScopes.cls
added tests/persistence/temp_table/undo_state/TestNoUndoTempTableMutations.cls
added tests/persistence/temp_table/undo_state/TestSharedTempTableUndoConflict.cls
added tests/persistence/temp_table/undo_state/TestStaticTempTableUndoAttribute.cls
added tests/persistence/temp_table/undo_state/TestSuite.cls
added tests/persistence/temp_table/undo_state/TestUndoAcrossProcedures.cls
added tests/persistence/temp_table/undo_state/TestUndoBlockForms.cls
added tests/persistence/temp_table/undo_state/TestUndoRecordIdentity.cls
added tests/persistence/temp_table/undo_state/TestUndoStateRedeclaration.cls
added tests/persistence/temp_table/undo_state/TestUndoableTempTableMutations.cls
added tests/persistence/temp_table/undo_state/UndoCalleeCreate.p
added tests/persistence/temp_table/undo_state/UndoCalleeCreateAndUndo.p
added tests/persistence/temp_table/undo_state/UndoCalleeCreateNoUndo.p
added tests/persistence/temp_table/undo_state/UndoProcDriver.p
added tests/persistence/temp_table/undo_state/UndoableRowsAfterUndo.p
added tests/persistence/temp_table/undo_state/zfile_set.txt
Committed revision 1874.
```

The failing test described in last note is TestUndoStateRedeclaration.SecondBufferOnNoUndoTableKeepsTheRow, FWD should not drop the record with id 2.

#52 - 08/20/2026 07:27 AM - Teodor Gorghe

Debugged into the test and the issue is related on how we flush temporary buffers when changing a field, inside transaction.

For temporary records, there is no dirty-share because they get immediately flushed to temp database on every change, except for one case: **inside a transaction**. Reference: TemporaryBuffer.isAutoCommit, Validation.validateMaybeFlush.
There is an issue on that: when a rollback happens, like undo, leave in this case, gets into RecordNursery.transactionEnded and newRecords/updatedRecords gets cleaned up. Nothing wrong related with undoable records, but it is an issue with no-undo records. RecordNursery forgets that it has a pending new/updated record, which triggers a missing/stale record on the next FOR EACH/FIND query.

This is what happens in the TestUndoStateRedeclaration.SecondBufferOnNoUndoTableKeepsTheRow.

Initial fix for this I think is to change the TemporaryBuffer.isAutoCommit is in transaction to be gated by the fact that the buffer is undoable.

#53 - 08/20/2026 07:42 AM - Alexandru Lungu

> There is an issue on that: when a rollback happens, like undo, leave in this case, gets into RecordNursery.transactionEnded and newRecords/updatedRecords gets cleaned up. Nothing wrong related with undoable records, but it is an issue with no-undo records. RecordNursery forgets that it has a pending new/updated record, which triggers a missing/stale record on the next FOR EACH/FIND query.

Mind that there is a (huge) change on #8388 about proper handling of UNDO in regard to RecordNursery. Any test that fails due to invalid state of records regarding dirty-share / RecordNursery due to UNDO is going to be fixed by #8388. Lets have the effort here (#11745) stick to the UNDO/NO-UNDO segregation of temp-tables with same schema.

#54 - 08/20/2026 07:44 AM - Teodor Gorghe

Ok, got it, but I have documented the issue here.

#55 - 08/20/2026 09:02 AM - Constantin Asofiei

Teodor: please get #8388, apply your changes over that branch and check the tests. It may be something very specific to no-undo temp-tables.

#56 - 08/21/2026 02:01 AM - Teodor Gorghe

I have applied the changes over 8388a and ran my test suite. The test from #11625 has been fixed, but the testcase described in #11745-52 not.

#57 - 08/21/2026 04:09 AM - Alexandru Lungu

> I have applied the changes over 8388a and ran my test suite. The test from #11625 has been fixed, but the testcase described in #11745-52 not.

What is the effect of this bug? I am inclined to prioritize 8388a merging and this task merging to fix multiple customer potential issues. The issue found in note 52 is empiric, but I want to understand how prone is to occur at a customer site. Is it a "Requested to persist DMO of type" error? If so, I

prefer to have it fixed, but on top of 8388a and 11745a changes (because these two change already quite a lot on the persistence layer and any fix should be implemented on top).

#58 - 08/21/2026 04:11 AM - Constantin Asofiei

Teodor Gorghe wrote:

> Initial fix for this I think is to change the TemporaryBuffer.isAutoCommit is in transaction to be gated by the fact that the buffer is undoable.

Yes, TemporaryBuffer.isAutoCommit is now !persistenceContext.isTransactionOpen(). But a NO-UNDO temp-table must also be auto-commit. Is changing this to consider NO-UNDO temp-tables enough?

#59 - 08/21/2026 04:14 AM - Teodor Gorghe

No, it is a different issue.
The recid conflict was fixed in this branch and the full description is in #11745-45.
The issue described yesterday is another issue, which is not fixed yet. I don't think it will cause recid conflicts because H2 knows about it and doesn't generate conflicting recids. So, just a phantom/leaking record in the H2.

#60 - 08/21/2026 06:20 AM - Constantin Asofiei

Teodor, about the 11745a_h2 rev 67:

- Table.setMultiplexNoUndo - this part can pose problems; if the entire table is no-undo, the someone trying to disable no-undo for a multiplex must not be allowed - I would fail there with SQLException than just return false.

```
public boolean setMultiplexNoUndo(int multiplex, boolean noUndo) {
   // the whole table is no-undo, so no multiplex of it can deviate
   if (isNoUndo) {
      return false;
   }
```

- **Alex** - please also review 11745a_h2.

#61 - 08/21/2026 06:23 AM - Teodor Gorghe

Table-wise no-undo is now set only when force-no-undo-temp-tables is enabled.

When that flag is disabled, it will use the setMultiplexNoUndo, with no undo flag on table-wise set to false.

#62 - 08/21/2026 06:28 AM - Constantin Asofiei

Teodor Gorghe wrote:

> Table-wise no-undo is now set only when force-no-undo-temp-tables is enabled.

But in H2 terms, someone trying to break the contract of the SQL-level no-undo state IMO must abend. I'd like to have that protection explicitly to avoid any possible mishap.

#63 - 08/21/2026 06:46 AM - Teodor Gorghe

Constantin Asofiei wrote:

> Teodor Gorghe wrote:
>
> > Table-wise no-undo is now set only when force-no-undo-temp-tables is enabled.
>
> But in H2 terms, someone trying to break the contract of the SQL-level no-undo state IMO must abend. I'd like to have that protection explicitly to avoid any possible mishap.

Ok, I understand, I am making this to throw error and display in the logs on FWD server as SEVERE.

#64 - 08/21/2026 06:49 AM - Constantin Asofiei

About 11745a: the only issue I see is about bufferType, which gets recreated in RecordBuffer.rebuildBufferType.

BufferType is used as a key in bufferManager.changeScopes. Switching undo state loses this information.

#65 - 08/21/2026 06:59 AM - Teodor Gorghe

Done the setMultiplexNoUndo abend in 11745a_h2/r68.

#66 - 08/21/2026 07:15 AM - Alexandru Lungu

> Alex - please also review 11745a_h2.

The changes look right to me. The critical invariants I want to make sure we respect:

- We don't change the no-undo flag in the middle of a transaction. All of the implementation relies on having the same state within a transaction, so an incorrect toggle of the no-undo flag while a transaction is being processed is fatal.\
- We don't leak records. This was the first and most important thing to keep in mind when working with this index. We had many times regressions on load testing due to records leaking (not being rollbacked correctly, recids not being dropped, multiplex not being closed, empty batches remaining, etc.)
- Recid management is a complete PITA. NO-UNDO does reclaim keys eagerly while UNOD reclaims keys after transaction ends. This is because an eventual UNDO can revive records that need to reclaim their original recid, so no new records can take that recid until the full transaction is committed. UPDATES make this a bit tricky because they are implemented as DELETE + CREATE, so CREATE should get the same recid as the one dropped by DELETE. Of course, the ROLLBACK of an UPDATE carries the same responsibility.
- There are some quirks regarding recid reclaiming. For instance, 4GL seems to store the records in pages and reclaiming happens only within the last page (or something like that if I recall correctly).
- FWD implements INSERT INTO SELECT FROM for COPY-TEMP-TABLE. Of course, such operations should respect recid, undo, and other invariants.
- FWD implements bulk DELETE of temp-tables. Of course, each individual delete of a bulk delete shall be registered to undo log and be sensitive to undo, recid reclaiming and other invariants.

Most importantly, we need (many serious tests) for each of the bullets above. The memory leak stuff shall be tested (either with heap dumps or debugging). We had in the past some memory leak tests (stress tests of many many hours) that failed due to leaks. We don't have access to them AFAIK anymore, but we can investigate on small tests.

Please also check other tasks that relate to FWD multiplexed index, recid reclaiming, etc. Make tests for them and retest in the original customer applications to ensure they are not regressed.

#67 - 08/21/2026 08:24 AM - Teodor Gorghe

Constantin Asofiei wrote:

> About 11745a: the only issue I see is about bufferType, which gets recreated in RecordBuffer.rebuildBufferType.
>
> BufferType is used as a key in bufferManager.changeScopes. Switching undo state loses this information.

Done in 11745a/r16706.

#68 - 08/21/2026 08:33 AM - Constantin Asofiei

Teodor Gorghe wrote:

> Done in 11745a/r16706.

Thanks. Let me know when you want 11745a_h2 to be merged to fwd_h2 so we can updated 11745a/build.xml and also publish the new fwd_h2 rev.

#69 - 08/21/2026 08:35 AM - Teodor Gorghe

Right now, I am looking for the previous H2 issues, to create testcases to test for leaks or regressions, but I will tell when testing is done.

#70 - 08/21/2026 09:06 AM - Constantin Asofiei

For the issue in #11745-45, do we postpone this in a different task/branch? I don't want to leave it as is.

#71 - 08/21/2026 09:08 AM - Constantin Asofiei

Constantin Asofiei wrote:

> For the issue in #11745-45, do we postpone this in a different task/branch? I don't want to leave it as is.

Sorry, I mean #11745-51, the SecondBufferOnNoUndoTableKeepsTheRow - does this test still fails?

#72 - 08/21/2026 09:17 AM - Teodor Gorghe

still fails.

#73 - 08/21/2026 09:25 AM - Constantin Asofiei

Teodor Gorghe wrote:

> still fails.

Does changing TemporaryBuffer.isAutoCommit to consider undoable state fix it?

#74 - 08/21/2026 09:28 AM - Teodor Gorghe

Constantin Asofiei wrote:

> Teodor Gorghe wrote:

> > > still fails.
>
> > Does changing TemporaryBuffer.isAutoCommit to consider undoable state fix it?
>
> Yes, it fixes it.

#75 - 08/21/2026 09:30 AM - Constantin Asofiei

Teodor Gorghe wrote:

> Constantin Asofiei wrote:
>
> > Teodor Gorghe wrote:
> >
> > > still fails.
> >
> > Does changing TemporaryBuffer.isAutoCommit to consider undoable state fix it?
>
> Yes, it fixes it.

Why not include this in this task? As is not related to #8388.

#76 - 08/21/2026 09:53 AM - Alexandru Lungu

I brought up #8388 because it embeds the fix for #6923 which is a massive rollback rework. If it is unrelated, I am ok to fix separately.

#77 - 08/24/2026 04:09 AM - Teodor Gorghe

Done the autocommit change in 11745a/r16707.
Also committed rev **69** in the H2 branch because I have found a small bug and a behavior in 4GL which we need to match.

About #11745-66, these commits should address the bugs found during testing.
I have made a test suite which creates lots of records, scenarios which flips the UNDO state, create/delete and updates, etc. I have took a heap dump before the run and after the run (while the session is alive), and I don't see any record which was being leaked into the H2. Only these are the only relevant retained instances:

```
2259:             1             64  org.h2.pagestore.db.MultiplexedScanIndex$Batch
2424:             2             48  com.goldencode.p2j.persist.BufferManager$BatchModeData
7420:             1             32  com.goldencode.p2j.persist.BufferType
```

#78 - 08/24/2026 04:14 AM - Teodor Gorghe

I am continuing to go through all H2 tasks and make the testcase (example: #10519)

#79 - 08/24/2026 06:02 AM - Teodor Gorghe

Added more testcases in **xfer** testcases. I think this now tests all what Alex pointed in #11745-66.
I don't see any obvious issue. Should I continue with testing on all active projects?

#80 - 08/24/2026 06:09 AM - Alexandru Lungu

Teodor/Constantin: can we have the H2 branch merged and a new fwd-h2 artifact released? This can ease the testing as we can simply plug the released artifact in the project.

From my POV we can move in Internal Test. I will try to deliver some tests as well in the process.

#81 - 08/24/2026 06:12 AM - Teodor Gorghe

I would have preferred to merge the H2 branch when all the testing is done. I don't want to go into a situation that I find a H2 issue and need to make a new branch while testing a project.

#82 - 08/24/2026 06:13 AM - Teodor Gorghe

Currently, I have temporary changed the build.gradle to take the H2 artifact from lib.local, to avoid the headaches while patching H2 artifact.

#83 - 08/24/2026 01:23 PM - Constantin Asofiei

Teodor Gorghe wrote:

> Currently, I have temporary changed the build.gradle to take the H2 artifact from lib.local, to avoid the headaches while patching H2 artifact.

Teodor, I can build 11745a_h2 and upload it to our gcd repo as 1.65-trunk, so build.gradle can be changed and target rev 1.65-trunk, and let testing move ahead without lib.client. If all testing passes, then 11745a_h2 can be merged (and re-release as rev 1.65-trunk again).

#84 - 08/25/2026 12:56 AM - Teodor Gorghe

Ok, I understand.
I'll update the build.gradle.

#85 - 08/25/2026 01:06 AM - Teodor Gorghe

build.gradle got updated in 11745a/r16708.

#86 - 08/25/2026 01:31 AM - Constantin Asofiei

- Status changed from Review to Internal Test

Teodor Gorghe wrote:

> build.gragle got updated in 11745a/r16708.

1.65-trunk was uploaded to redmine artifacts. We can go ahead with testing.

#87 - 08/25/2026 06:52 AM - Alexandru Lungu

I created some tests, committed to testcases/1883:

- tests/persistence/temp_table/undo_state/dmo_identity
- tests/persistence/temp_table/recid_undo_log_compound_ops

They pass both with trunk and 11745a. There is however a test that fails with both:
com.goldencode.testcases.tests.persistence.temp_table.recid_undo_log_compound_ops.TestBulkDeleteUndoLog

The tests check if undoing a empty-temp-table is actually re-populating the temp-table. In OE it does, but in FWD it does not. I will let me know if I can have a quick look and check if this is a FWD issue that can be quickly fixed in FWD 11745a. I am worried that such scenario can also trigger DMO recid conflict (a bulk delete that is not correctly rollbacked may mess with recid) which may be the main culprit of #11237. WIP

#88 - 08/25/2026 07:02 AM - Teodor Gorghe

Ok, might not be related with that test, but still with empty-temp-table, but when the temp-tables are no-undo. May be something which is related to the force undo flag from directory.

#89 - 08/25/2026 08:14 AM - Alexandru Lungu

Teodor Gorghe wrote:

> Ok, might not be related with that test, but still with empty-temp-table, but when the temp-tables are no-undo. May be something which is related to the force undo flag from directory.

This was a very quirky thing. Apparently, EMPTY-TEMP-TABLE is not a method for a temp-table handle. Although it is documented like such, actually running it yields a warning "EMPTY-TEMP-TABLE is not a queryable attribute" (even if the syntax is htt:empty-temp-table(). method-like). I did not

spot this because it was wrapped in a no-error clause. In FWD it actually worked at conversion time, so hBulk.unwrapEmptyTempTable().deleteAll() was generated. I rewrote the statement using a buffer receiver and made it work.

This is a bit stressful because I think I spotted empty-temp-table being used on a temp-table handle instead of a buffer handle in customer application. But at the same time I am not certain if this method works only in some OE versions. Teodor/Constantin, can you confirm my speculation (running empty-temp-table() on a temp-table handle actually yields a warning and the table is not cleared)?

#90 - 08/25/2026 09:02 AM - Teodor Gorghe

Yes, that's indeed true. That happens on OE 11.6.

```
def temp-table ttA no-undo
    field f1 as int.
def buffer bA for ttA.

temp-table ttA:empty-temp-table.

MESSAGE "done".
```

buffer ttA:empty-temp-table works, but temp-table ttA:empty-temp-table not.

#91 - 08/25/2026 09:41 AM - Constantin Asofiei

Teodor: please look into the history of persist.TempTable:

- 018 OM 20190513 Extends EmptyTempTable to access EMPTY-TEMP-TABLE method.

Ovidiu: do you recall why this was added?

#92 - 08/25/2026 09:54 AM - Ovidiu Maxiniuc

The bzr log for revno is 11310.1.8:

```
revno: 11310.1.8
author: Ovidiu Maxiniuc <om@goldencode.com>
committer: Constantin Asofiei <ca@goldencode.com>
branch nick: 3751a
timestamp: Mon 2019-05-13 19:22:38 +0300
message:
  Extracted EMPTY-TEMP-TABLE in a separate interface and extended both TempTable and Buffer from it. refid: #3
809, #3751
```

Feature #3809 (ProDataSet support) is obvious (cleaning the tables/datasets), but Feature #3751 (implement support for OO 4GL and structured error handling) not. At least not directly.

#93 - 08/25/2026 09:59 AM - Ovidiu Maxiniuc

Well, after more careful analysis I see that the branch was 3751a which explains the reference to #3751.
But why this particular update was committed to 3751a and not to 3809x is still an enigma .

#94 - 08/25/2026 10:13 AM - Constantin Asofiei

Alex/Teodor: is empty-temp-table a concern for this task? It may be that we emitted for each tt1: delete tt1. end. via a tempTable(tt1).emptyTempTable() at some point, instead via the buffer. And is just something that got missed. Anyway, we should fix it, but in a different task.

#95 - 08/25/2026 12:29 PM - Teodor Gorghe

I don't think it is a concern from POV.

I have tried to reproduce the issues from #11237 & #11632, but I wasn't able to. AFAIK, both points to empty-temp-table and deleteAll, with force-no-undo-temp-tables set to **true**.

#96 - 08/26/2026 04:04 AM - Alexandru Lungu

> I have tried to reproduce the issues from #11237 & #11632, but I wasn't able to. AFAIK, both points to empty-temp-table and deleteAll, with force-no-undo-temp-tables set to true.

I will request directory.xml again for the systems in #11237 and #11632. Maybe the force-no-undo was removed. I recall that we had a talk about it (last year maybe?) and the customer in question had second thoughts about it. So they might have considered testing without it on some environments. Low chances tho.

> Alex/Teodor: is empty-temp-table a concern for this task? It may be that we emitted for each tt1: delete tt1. end. via a tempTable(tt1).emptyTempTable() at some point, instead via the buffer. And is just something that got missed. Anyway, we should fix it, but in a different task.

No, but please open a separate DB task. I think this is an easy scenario to miss, especially if you use no-error (like I did). It stresses me that I found several cases in customer source code where a table-handle is actually used with EMPTY-TEMP-TABLE.

#97 - 08/26/2026 06:01 AM - Constantin Asofiei

Alex, please open a task in the customer's project and ask why that code has h:empty-temp-table(). for a temp-table handle. I just want to make sure there isn't some kind of weird quirk in 4GL.

#99 - 08/26/2026 08:50 AM - Alexandru Lungu

I will let #11795 drive any decision on empty-temp-table over temp-table handles. From my POV the testing is clear on #11745.
Teodor, is there other testing to be done here?

#100 - 08/26/2026 08:51 AM - Teodor Gorghe

Started to run conversion and runtime testing on all projects.
For now, 11745a testing has passed on at least 3 projects.

#101 - 08/27/2026 01:14 AM - Teodor Gorghe

There is an unexpected failure on ChUI regression tests, caused by **11745a** changes.
Currently investigating.

#102 - 08/27/2026 02:06 AM - Teodor Gorghe

Alex, it is a bit unrelated to this task, but the changes for zfile_set.txt on **testcases** project are intentional (make on rev 1850 and 1883)?

#103 - 08/27/2026 02:28 AM - Teodor Gorghe

The failure from ChUI regression testing is related to the Paul's fix about the no-undo flag. The issue was already fixed by Octavian, well documented in #3211-293.

I will rebase the branch, rerun the conversion and check if there are any more fails.

#104 - 08/27/2026 02:48 AM - Teodor Gorghe

Rebased task branch **11745a** to trunk rev **16724**. Last revision is now **16730**.

#105 - 08/27/2026 03:15 AM - Teodor Gorghe

Redid the conversion for ChUI regression and the SharedVariableManager.addTempTable undoable flag is now correct.
Runtime testing is ongoing.

#106 - 08/27/2026 06:46 AM - Teodor Gorghe

ChUI Regression testing has passed.

#107 - 08/27/2026 09:14 AM - Alexandru Lungu

Anything left to test?

#108 - 08/27/2026 09:15 AM - Teodor Gorghe

4 projects left to test. I will tell when the testing on projects is done.

#109 - 08/27/2026 09:19 AM - Teodor Gorghe

Didn't documented in the testing plan, but I have asked to Dănuț to run conversion + runtime testing on POC, I don't know how is going on, but I am mostly interested if the conversion passes.

#110 - 08/28/2026 04:05 AM - Artur Școlnic

Tested the multi tenant project.

Conversion has differences like

```
<    tt.Buf tt = (tt.Buf) TemporaryBuffer.useShared("tt", tt.Buf.class, false);
---
>    tt.Buf tt = (tt.Buf) TemporaryBuffer.useShared("tt", tt.Buf.class);
```

it finished successfully.

At runtime there is an issues, in the server log I have multiple severe entries like

```
26/08/28 01:56:16.415-0600 |  SEVERE | com.goldencode.p2j.util.ErrorManager | ThreadName:MSA Worker #24 for ap
p appsrv, Session:00000002, ThreadId:00000009, User:appsrvprocess | In procedure proc.p, shared temp-table tt
has a conflict in field, index or undo status. (2075)
```

this causes some scenarios to fail.
Please let me know if this is cause by a config I missed or a legitimate issues.

#111 - 08/28/2026 04:06 AM - Teodor Gorghe

Did you used the latest **11745a**, including the rebase to the trunk rev **16724**?

#112 - 08/28/2026 04:07 AM - Artur Şcolnic

Used rev 16730 on top of trunk 16724.

#113 - 08/28/2026 04:08 AM - Teodor Gorghe

Investigating right now. Seems like the problem is on the producer side, **SharedVariableManager.addTempTable**, and #1752 didn't fixed your case.

#114 - 08/28/2026 04:10 AM - Teodor Gorghe

Can you upload the conversion (**src** is only needed) on devsrv01 /tmp, so I can take a look on the converted source code?

#115 - 08/28/2026 04:11 AM - Teodor Gorghe

Or perhaps, you took the patch and applied over 10614_main.

#116 - 08/28/2026 04:12 AM - Artur Şcolnic

I used 11745a.

#117 - 08/28/2026 04:14 AM - Teodor Gorghe

Ok, it is not a false alert. I will wait for the converted sources.

#118 - 08/28/2026 04:15 AM - Artur Școlnic

Uploaded ..._src_11745a.zip to tmp. To replicate the issues, just run harness.

#119 - 08/28/2026 05:40 AM - Teodor Gorghe

Artur, I don't have an explanation why this issue occurs. I know that there is a conflict between two temp-tables with the same definition and that is how it also happens on 4GL if the undoable flag doesn't match.

I don't have the cvt files and the jar just yet, but as I can see on the converted sources, they are right.
Please make sure that you actually applied the converted jar, because using an old jar could also cause this issue.

#120 - 08/28/2026 06:02 AM - Artur Școlnic

Yes, that was it, I forgot to replace the application and h2 jars. Retested and it seems fine, sorry for that.

#121 - 09/01/2026 03:28 AM - Constantin Asofiei

- Status changed from Internal Test to Merge Pending

Please merge 11745a and 11745a_h2 now.

#122 - 09/01/2026 03:40 AM - Teodor Gorghe

- Status changed from Merge Pending to Test

Branch **11745a** was merged into trunk as rev. **16731** and archived.
Branch **11745a_h2** was merged into FWD-H2 **trunk** as rev **65** and archived.

Files

fwd-h2-11745.patch	13.6 KB	08/18/2026	Teodor Gorghe