Conversion Tools - Bug #12106

Temp-table queries lose their index because the cached temp-table P2O lookup is built before P2O post-processing adds the indexes

11/03/2026 02:33 AM - Octavian Adrian Gavril

Status:	Review	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Octavian Adrian Gavril	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**reviewer:**	Constantin Asofiei, Dănuț Filimon
vendor_id:	GCD	**production:**	No
case_num:		**env_name:**	
version_reported:		**topics:**	Incremental Conversion
version_resolved:			

Description

History

#1 - 11/03/2026 02:42 AM - Octavian Adrian Gavril

When a conversion run converts a single file, the queries on its temp-tables can lose their index, even with USE-INDEX. The temp-table lookup cached by P2OLookup.loadTemporarySchema() for that file holds no indexes. The same problem was found in #7169-69.

The testcase is tt_useidx.p:

```
define temp-table tt no-undo
   field f1 as integer
   field f2 as character
   index idx-f1 is primary f1
   index idx-f2 f2.

for each tt use-index idx-f2:
   display tt.f1 tt.f2.
end.
```

To reproduce it on trunk **16822**:

1. Add tt_useidx.p to a project with a few other programs and run a full conversion. The query in TtUseidx.java is correct: query0.initialize(tt, ((String) null), null, "tt.f2 asc").
2. Add one space to tt_useidx.p and run an incremental conversion. Only tt_useidx.p is converted again.
3. The query in TtUseidx.java is now wrong: query0.initialize(tt, ((String) null), null, "tt.recid asc").

A full conversion of tt_useidx.p alone gives the same wrong query.

The lookup of tt_useidx.p is rebuilt when "P2O Post-processing (temp-table schema files)" visits it, as its .p2o AST exists for the first time. The INDEX nodes are added by schema/p2o_post.xml only after that visit. isCurrentLookupFor() checks only the artifact, the presence of the AST and the class definition, so all later phases reuse this lookup without indexes. When other files are converted in the same run, visiting them replaces the cached lookup, so tt_useidx.p is usually reloaded with its indexes.

The flaw is older than the reuse check added in **16737**: the P2OLookup.java of **16736** on top of trunk **16822** gives the same wrong query. The reuse check must notice that the P2O AST changed after the lookup was built.

#2 - 11/03/2026 02:42 AM - Octavian Adrian Gavril

- Assignee set to Octavian Adrian Gavril

- Status changed from New to WIP

I'm looking into this one.

#3 - 11/03/2026 06:07 AM - Octavian Adrian Gavril

- % Done changed from 0 to 100

- Status changed from WIP to Review

- reviewer Constantin Asofiei, Dănuț Filimon added

I've committed a fix for this in **12106a**/**16827**.

The reason why the queries lose their index is the order in which the temp-table lookup is built and the P2O AST is completed. P2OLookup.loadTemporarySchema() keeps the last lookup it built and reuses it as long as isCurrentLookupFor() accepts it. When P2O Post-processing (temp-table schema files) visits a file, its .p2o AST exists for the first time, so the lookup is rebuilt. Only after that visit does the phase add the INDEX nodes and save the AST again. The check compares the artifact, the presence of the AST and the class definition, so it cannot see the second save, and every later phase reuses the lookup without indexes. The lookup is cached per conversion thread, so with several threads the problem appears at random, also in runs with many files.

The fix makes the reuse check notice every save of a P2O AST:

- P2OLookup counts the P2O AST saves in an AtomicLong shared by all threads.
- DataModelWorker.persistTarget() increments the counter right after saveTree().
- A temp-table lookup records the counter first thing in its constructor, before it loads its AST and the ASTs of the parent classes.
- isCurrentLookupFor() also rejects a lookup whose recorded value is no longer the current one.

Saves happen only in the P2O phases, so the lookup is still reused in all later phases. At runtime persistTarget() is not called, so the counter stays 0 and dynamic temp-tables behave as before.

The cost on a large project is what the testcases cannot show. Each save makes every thread reload its lookup during the two P2O phases. On the testcases the conversion times stay within the run-to-run noise.

Danut, Constantin, please review the changes.

#4 - 11/03/2026 09:50 AM - Octavian Adrian Gavril

This issue is already fixed in **7169a**, in revision **16833**. After each conversion pass, TransformDriver.endPass() calls P2OLookup.resetTemporarySchema() on the main thread and on every conversion thread. The next pass then rebuilds the temp-table lookup from the completed P2O AST, which has the indexes.

I tested **7169a** revision **16835** with twelve conversion scenarios which check the sort clauses generated for the queries. They cover tt_useidx.p converted alone, an incremental conversion after a space is added, an incremental conversion which adds an index, FIND ... USE-INDEX, CAN-FIND and implicit index selection, a temp-table LIKE another one, a temp-table defined in an include file, a NEW SHARED and SHARED pair, and a chain of three classes with inherited and own temp-tables.

	1 thread	4 threads
trunk **16826**	8 of 12 fail with recid asc	1, 3 and 1 failures in three runs
7169a 16835	12 of 12 pass	12 of 12 pass in each of three runs

The open question is whether we want this fix in trunk sooner, in a dedicated branch, or whether it can stay in **7169a** and #12106 can be rejected.