Database - Bug #12210

Speed up field access through BUFFER parameters

11/17/2026 07:37 AM - Artur Școlnic

Status:	Test	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Artur Școlnic	**% 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 - Feature #9976: eliminating DMO usage in inter-program l...	**New**

History

#1 - 11/17/2026 07:37 AM - Artur Școlnic

- Related to Feature #9976: eliminating DMO usage in inter-program linkage added

#2 - 11/17/2026 07:41 AM - Artur Școlnic

- Status changed from New to WIP

- Assignee set to Artur Școlnic

Issue

Field access through a BUFFER parameter was slow in two cases, both caused by the proxy layers discussed in #9976:

- **Different DMO:** when the callee's temp-table differs from the caller's only in non-structural attributes (e.g. labels), the argument is wrapped by RecordBuffer.createProxy(Class, Object), whose handler called getMethod() on every field access. On a BufferImpl proxy class that scans hundreds of public methods.
- **Nested parameters:** each level a buffer is passed down added one more ArgumentBuffer proxy on top of the previous one, so every field access paid one reflective hop per level, and the cost grew faster than linearly with depth.

Changes

- RecordBuffer.createProxy(Class, Object) caches the resolved argument method per argument class (ClassValue), so getMethod() runs once per method instead of once per call.
- RecordBuffer.makeArgumentBuffer(): if the incoming buffer is already an argument buffer, the new one wraps what that one wraps, so a buffer passed down any number of levels stays a single proxy deep. Behaviour is unchanged, since only the innermost argument buffer's name and alias ever took effect.
- ProxyFactory.getInvocationHandler() (new) returns the handler of a proxy's own, outermost proxy class; its cache is cleared by removeProxy() like the other per-class caches.

Testing

End-to-end 4GL benchmarks (batch client, same conversion and server configuration for every build, only p2j.jar differing):

- bench.p RUNs a callee with BUFFER tt whose temp-table is identical (same DMO) or differs only in labels (different DMO);
- bench_depth.p passes BUFFER b down 1 to 20 levels of a recursive internal procedure;
- each case reads b.f1 200,000 times; the values are medians in ms, excluding the warm-up round.

Scenario	Clean trunk	Cached method only	Unwrap only	Both
Same DMO, depth 1	15	15	18	17

Different DMO	**1020**	**21**	1103	**27**
Depth 1	20	18	19	18
Depth 2	31	29	**19**	**20**
Depth 5	94	73	**19**	**20**
Depth 10	299	211	**22**	**22**
Depth 20	**1139**	972	**24**	**23**

With both changes, a field read through a different-DMO parameter is about 40 times faster, and through a buffer passed 20 levels deep about 50 times faster; all scenarios now cost the same as the plain same-DMO case. Each build ran on a separate server start and the same build varies by up to about 20% between runs, so only the large differences are significant. All checksums were correct in every run.

I will do some regression testing.
Constantin, can you please review 12210a.

#3 - 11/17/2026 07:41 AM - Artur Școlnic

- Status changed from WIP to Review

- % Done changed from 0 to 100

- reviewer Constantin Asofiei added

#4 - 11/17/2026 07:54 AM - Constantin Asofiei

Artur, nice find. It may be worth it to check other getMethod or getField non-cached calls.

Otherwise, about makeArgumentBuffer - this was needed originally to translate FQLs which use the definition alias (buffer name), to use the argument's alias. If this still works, then the changes are good.

#5 - 11/17/2026 09:26 AM - Artur Școlnic

makeArgumentBuffer / FQL alias

Still works. Tested with a buffer passed down 5 levels under different names (internal procedures, plus an external procedure with a different-DMO temp-table in the middle), running FIND, a self-join, CAN-FIND and a failing FIND against each level's own name. Output is identical on clean trunk and with the changes. This is expected: overrideDMOAlias is first-wins, so only the innermost alias ever took effect.

Other non-cached reflection lookups

I cached the 7 most frequently executed ones and measured each separately (ms, median):

#	Lookup	Benchmark	Before	After	Gain
1	Utils.invoke setAccessible	5M field reads	243	236	none
2	DMO constructor (SQLQuery, Loader, ...)	5,000-row FOR EACH x20	165	164	none
3	trigger constructor	2,000 CREATEs with WRITE trigger	2698	2427	noise

4	getConstructor in RUN parameter matching	200k mismatched RUNs	490	464	noise
5	NEW constructor in ObjectOps	100k NEW	434	422	none
6	**RecordBuffer.copyCommonFields**	**5,000 MARK-ROW-STATE**	**347**	**218**	**~1.6x**
7	**Call INVOKE method lookup**	**100k INVOKE (get / set attr)**	**651 / 660**	**31 / 45**	**~15-20x**

Only lookups by method name on large classes (getMethod, getMethods, misses throwing NoSuchMethodException) are expensive; constructors and setAccessible are already cached by the JDK. I implemented 6 and 7, please take a look.

#6 - 11/17/2026 09:35 AM - Constantin Asofiei

- Status changed from Review to Internal Test

Lets get this tested then. Do a sweep for other getMethod/Field uncached in FWD please and see if there are other points we can optimize. Thanks.

#7 - 11/18/2026 03:23 AM - Artur Școlnic

I added just one more improvement, each NEW now takes the CLASS EVENT signature from a per-class cache instead of rebuilding it by reflection, which up to now threw one exception for every class between the object's class and the one declaring the event; NEW is about 1.4× faster, and about 2.6× for subclasses.

Multi tenant app, large gui app unit tests, chui regression and etf passed.

#8 - 11/18/2026 07:05 AM - Artur Școlnic

Can we merge this one?

#9 - 11/18/2026 07:06 AM - Constantin Asofiei

- Status changed from Internal Test to Test

Yes, go ahead.

#10 - 11/18/2026 07:11 AM - Artur Școlnic

12210a was merged to trunk/16857.