Database - Bug #11992

[Regression] Inverted param < field / param <= field comparison after the trunk/r16278 FQLPreprocessor rewrite.

10/14/2026 04:51 AM - Stefanel Pezamosca

Status:	Test	**Start date:**	
Priority:	Urgent	**Due date:**	
Assignee:	Stefanel Pezamosca	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**reviewer:**	Alexandru Lungu, Eduard Soltan
vendor_id:	GCD	**production:**	No
case_num:		**env_name:**	
version_reported:		**topics:**	
version_resolved:			

Description

History

#2 - 10/14/2026 04:55 AM - Stefanel Pezamosca

Regression from **trunk/r16278** (10729a merge), see #10729-149.

When FQLPreprocessor rewrites param < field / param <= field, it swaps the operands but keeps the operator. The query then runs field < param.

Example: indexed tt.f1 with values 1, 2, ? and v = 1:

WHERE	OE	FWD
v < tt.f1	2 ?	?
v <= tt.f1	1 2 ?	1 ?

Proposed fix: keep the original operand order when rebuilding the comparison.

```
=== modified file 'src/com/goldencode/p2j/persist/FQLPreprocessor.java'
@@ -6138,10 +6139,8 @@
-                HQLAst oppositeOperand = (HQLAst) ((next.getIndexPos() == 0)
-                                                   ? next.getNextSibling()
-                                                   : next.getPrevSibling());
-
+                boolean substFirst = (next.getIndexPos() == 0);
+
               HQLAst oppositeOperand = (HQLAst) (substFirst ? next.getNextSibling() : next.getPrevSibling());

                 oppositeOperand.remove();
@@ -6153,8 +6152,9 @@
                 next.remove();

-                compareOperator.addChild(oppositeOperand);
-                compareOperator.addChild(next);
+                // keep the operand order, the operator is not mirrored
+                compareOperator.addChild(substFirst ? next : oppositeOperand);
+                compareOperator.addChild(substFirst ? oppositeOperand : next);
                 oppositeOperand.setParent(compareOperator);
                 next.setParent(compareOperator);
```

This fixes tests/query/TestJoinQueries: testOuterLessThanInner, testOuterLessterLessThanInner.

Comparisons against the unknown value itself are wrong both with and without the patch, so they need a separate fix: ? > f1, ? >= f1, f1 < ?, f1 <= ?,

and v < f1 / v > f1 when v = ?.

#3 - 10/14/2026 05:00 AM - Alexandru Lungu

- Assignee set to Stefanel Pezamosca

Added Artur as a watcher.

#4 - 10/15/2026 04:24 AM - Stefanel Pezamosca

- Status changed from New to WIP

- % Done changed from 0 to 100

Created **11992a** from trunk **r16804** and committed a better fix than #11992-2 in **rev. 16805**.

Instead of rebuilding the comparison, FWD now keeps the original one exactly as written and just wraps it with the null check:

```
(<original comparison> or field is null)            -- IS_NULL * 1000 + LT / LTE / GT / GTE
(<original comparison> and field is not null)       -- NOT_NULL * 1000 + EQUALS
```

Why the old code was unnecessary?

- The rebuilt comparison always had the same operator as the original one, because this code is only reached after the original operator has been checked. Rebuilding it added nothing.
- It did take a lot of work: creating a new node, setting its operator text again, moving both operands into it, and turning the old node into the is null check. Moving the operands is exactly where they got swapped.
- Keeping the original comparison means the operator and the operand order are always right, so this mistake can't happen again. The code is also about 60 lines shorter.
- Nothing later in the processing depended on the field being on the left, and the generated SQL has the same shape as before (only the operand order changes), so index use is not affected.

I ran the full ABLUnit suite I use to test other branches (3303 tests). There are no regressions, and TestJoinQueries testOuterLessThanInner, testOuterLessOrEqualInner and testDynamicOuterLessThanInner now pass.

Not covered:
Comparisons against the unknown value itself (? > f1, f1 < ?, or v < f1 when v = ?) behave the same as on trunk. They seem related to what #11108 is working on.

#5 - 10/15/2026 04:24 AM - Stefanel Pezamosca

- Status changed from WIP to Review

#6 - 10/15/2026 04:47 AM - Alexandru Lungu

- reviewer Alexandru Lungu, Eduard Soltan added

#7 - 10/15/2026 06:34 AM - Alexandru Lungu

I am ok with the changes, but I think there are still some cases that are left unfixed. These come from two places in FQLPreprocessor.simplifyUnknownsOrInjectNullCheck:

1. f > ? and ? < f: for an indexed field the code picks the IS_NULL*1000+GT / +LT rewrite without checking whether the parameter is unknown. OE returns nothing above unknown, so with an unknown parameter the result should be false.
2. f < ? and ? > f: the code first picks field is not null, which matches OE, then replaces it with true when the field is indexed. That contradicts the method's own javadoc, which says t.f < ? is equivalent to t.f <> ?.

Also, please test tables without indexes (e.g. queries that end up with tt.recid asc in the order by).

#8 - 10/15/2026 07:26 AM - Eduard Soltan

Stefanel Pezamosca wrote:

> **Not covered:**
> Comparisons against the unknown value itself (? > f1, f1 < ?, or v < f1 when v = ?) behave the same as on trunk. They seem related to what #11108 is working on.

Stefanel, I rebased 11108c. Can check these cases with the 11108c?

#9 - 10/15/2026 07:30 AM - Alexandru Lungu

PS:

> Comparisons against the unknown value itself (? > f1, f1 < ?, or v < f1 when v = ?) behave the same as on trunk. They seem related to what #11108 is working on.

My bad, I did not spot these. I am ok with the changes considering the limitations.

#10 - 10/15/2026 08:52 AM - Stefanel Pezamosca

- reviewer deleted (Alexandru Lungu, Eduard Soltan)

Eduard Soltan wrote:

> Stefanel Pezamosca wrote:
> Stefanel, I rebased 11108c. Can check these cases with the 11108c?

It fixes some cases, not all, but also seem to have some other regressions maybe something that got removed after the rebase, I'm not sure.

ExampleExample

1. Unknown value against an indexed field: still wrong on 11108c

```
DEFINE TEMP-TABLE tt FIELD f1 AS INTEGER INDEX i1 IS PRIMARY f1.
DEFINE VARIABLE v AS INTEGER   NO-UNDO INITIAL ?.
DEFINE VARIABLE s AS CHARACTER NO-UNDO.

CREATE tt. tt.f1 = 1.
CREATE tt. tt.f1 = 2.
CREATE tt. tt.f1 = ?.

s = "". FOR EACH tt WHERE ? >= tt.f1 BY tt.f1: s = s + " " + STRING(tt.f1). END.
MESSAGE "? >= f:" s.     /* OE: 1 2 ?   11108c: (none) */
s = "". FOR EACH tt WHERE tt.f1 <= ? BY tt.f1: s = s + " " + STRING(tt.f1). END.
MESSAGE "f <= ?:" s.     /* OE: 1 2 ?   11108c: (none) */
s = "". FOR EACH tt WHERE v > tt.f1 BY tt.f1: s = s + " " + STRING(tt.f1). END.
MESSAGE "v=? > f:" s.    /* OE: 1 2     11108c: 1 2 ? */
s = "". FOR EACH tt WHERE tt.f1 < v BY tt.f1: s = s + " " + STRING(tt.f1). END.
MESSAGE "f < v=?:" s.    /* OE: 1 2     11108c: 1 2 ? */
```

2. Known value: correct on trunk, wrong on 11108c

```
DEFINE TEMP-TABLE tt FIELD f1 AS INTEGER INDEX i1 IS PRIMARY f1.
DEFINE TEMP-TABLE ta FIELD f1 AS INTEGER INDEX ia IS PRIMARY f1.
DEFINE TEMP-TABLE tb FIELD f1 AS INTEGER INDEX ib IS PRIMARY f1.
DEFINE VARIABLE v AS INTEGER   NO-UNDO INITIAL 1.
DEFINE VARIABLE s AS CHARACTER NO-UNDO.
DEFINE VARIABLE i AS INTEGER   NO-UNDO.

CREATE tt. tt.f1 = 1.
CREATE tt. tt.f1 = 2.
CREATE tt. tt.f1 = ?.
DO i = 1 TO 3:
   CREATE ta. ta.f1 = i.
   CREATE tb. tb.f1 = i.
END.
CREATE tb. tb.f1 = ?.

s = "". FOR EACH tt WHERE tt.f1 > v BY tt.f1: s = s + " " + STRING(tt.f1). END.
MESSAGE "f > v:" s.      /* OE: 2 ?     11108c: 2 */
s = "". FOR EACH tt WHERE tt.f1 >= v BY tt.f1: s = s + " " + STRING(tt.f1). END.
MESSAGE "f >= v:" s.     /* OE: 1 2 ?   11108c: 1 2 */
s = "". FOR EACH ta, EACH tb WHERE tb.f1 > ta.f1: s = s + "(" + STRING(ta.f1) + "," + STRING(tb.f1) + ")". END
.
MESSAGE "b > a:" s.      /* OE: (1,2)(1,3)(1,?)(2,3)(2,?)(3,?)   11108c: (1,2)(1,3)(2,3) */
s = "". FOR EACH ta, EACH tb WHERE tb.f1 >= ta.f1: s = s + "(" + STRING(ta.f1) + "," + STRING(tb.f1) + ")". EN
D.
MESSAGE "b >= a:" s.     /* OE: (1,1)(1,2)(1,3)(1,?)(2,2)(2,3)(2,?)(3,3)(3,?)   11108c: (1,1)(1,2)(1,3)(2,2)(2
,3)(3,3) */
```

So we can go with 11992a as it is, and come back to the other issues later, right?
The testcases used here are covering most of the cases so I don't know if there are other tests that I should do, Alex ? If not, I think 11992a is pretty safe.

#11 - 10/15/2026 08:53 AM - Stefanel Pezamosca

- reviewer Alexandru Lungu, Eduard Soltan added

#12 - 10/15/2026 08:59 AM - Alexandru Lungu

- Status changed from Review to Internal Test

- Priority changed from Normal to Urgent

> So we can go with 11992a as it is, and come back to the other issues later, right?

I agree. I think the issues discovered here are quite urgent as it makes some queries return other records than expected completely. Please proceed with testing and lets merge this asap.

#13 - 10/15/2026 08:59 AM - Alexandru Lungu

PS: we will need to find out the customer branches that are susceptible to this bug and suggest porting this fix.

#14 - 10/16/2026 07:40 AM - Stefanel Pezamosca

I tested **11992a rev. 16805** against OpenEdge and against trunk **r16804** (11992a with the fix reverse-applied), using the same probes on all three. I think **11992a is safe to merge**.

Coverage:

- Field types: integer, character, date and decimal.
- Indexes: ascending, descending, two-field (leading equality, non-leading component), unique, and a table with no index.
- Comparisons: all 8 v op f / f op v forms with <, <=, > and >=, plus = and <>, with variables and literals, combined with AND/OR.
- Statements: FOR EACH with and without BY, FIND FIRST/LAST/NEXT/PREV, CAN-FIND, and joins (EACH, FIRST, LAST).
- Dynamic queries: GET-NEXT/GET-PREV and buffer FIND-FIRST/FIND-LAST.

#15 - 10/16/2026 07:43 AM - Alexandru Lungu

- Status changed from Internal Test to Merge Pending

Please merge 11992a to trunk now.

> PS: we will need to find out the customer branches that are susceptible to this bug and suggest porting this fix.

Please also assess this.

#16 - 10/16/2026 08:00 AM - Stefanel Pezamosca

- Status changed from Merge Pending to Test

Branch 11992a was merged into trunk as rev. 16808 and archived.

> PS: we will need to find out the customer branches that are susceptible to this bug and suggest porting this fix.

I'll check.

#17 - 10/16/2026 08:26 AM - Stefanel Pezamosca

These branches have the bug: 10614_main, 8755d, 9866e (d?), 11748a.

The 11992a commit needs createSubtree() (trunk r16787+), so we need to use this minimal patch instead. It applies cleanly to all four, and on trunk r16804 it gives the same results as 11992a.

PatchPatch

```
--- a/src/com/goldencode/p2j/persist/FQLPreprocessor.java
+++ b/src/com/goldencode/p2j/persist/FQLPreprocessor.java
@@ -6731,10 +6731,8 @@
                        break;
                }

-               HQLAst oppositeOperand = (HQLAst) ((next.getIndexPos() == 0)
-                                                  ? next.getNextSibling()
-                                                  : next.getPrevSibling());
-
+               boolean substFirst = (next.getIndexPos() == 0);
+
               HQLAst oppositeOperand = (HQLAst) (substFirst ? next.getNextSibling() : next.getPrevSibling());

                oppositeOperand.remove();
                HQLAst copyOppositeOperator = (HQLAst) oppositeOperand.duplicate();
@@ -6746,8 +6744,9 @@

                next.remove();

-               compareOperator.addChild(oppositeOperand);
-               compareOperator.addChild(next);
+               // keep the operand order, the operator is not mirrored
+               compareOperator.addChild(substFirst ? next : oppositeOperand);
+               compareOperator.addChild(substFirst ? oppositeOperand : next);
                oppositeOperand.setParent(compareOperator);
                next.setParent(compareOperator);
```

#18 - 10/19/2026 08:50 AM - Alexandru Lungu

- 10614_main: Please open a task in customer project and describe the bug. I send you by email the process for the customer.
- 8755d and 11748a: Please port to these task branches now.
- 9866e: Radu, please assess.

#20 - 10/20/2026 04:16 AM - Radu Apetrii

Alexandru Lungu wrote:

> - 9866e: Radu, please assess.

I agree that this is an important fix, so I added it to the #11981 list.