Database - Bug #12144

H2 joins on fields that can be unknown scan both tables instead of using an index

11/09/2026 06:28 AM - Stefanel Pezamosca

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

Description

History

#1 - 11/09/2026 06:59 AM - Stefanel Pezamosca

- % Done changed from 0 to 100

- Status changed from New to WIP

- reviewer Alexandru Lungu added

On H2, a join on two fields that can be unknown now uses an index instead of comparing every pair of records. Joining 4000 records with 8000 child records went from 3.8-6.1 s to 27-42 ms, with the same results.

Problem

In the 4GL, two unknown values are equal, so FQLPreprocessor turns t1.f1 = t2.f2 into:

```
(t1.f1 = t2.f2 or (t1.f1 is null and t2.f2 is null))
```

H2 cannot use an index for this or form, so it reads both tables in full and checks every pair of rows. This affects every temp-table join of this shape, and joins on permanent tables in an H2 database. EXPLAIN of FOR EACH order, EACH order-line WHERE order-line.order-num = order.order-num:

```
Without the change:  order_      tableScan
                     order_line  tableScan
With the change:     order_line  tableScan
                     order_      idx__order__order_num: ORDER_NUM IS NOT DISTINCT FROM ORDERLINE_1_.ORDER_NUM
```

Fix

On H2, FqlToSqlConverter.rewriteNullSafeEquals() writes the or form as t1.f1 IS NOT DISTINCT FROM t2.f2, which H2 can serve from an index. Dialect.getIndexableNullSafeEquals() returns null by default, so the other dialects are not affected.

The two forms differ only when exactly one value is unknown: the or form gives null, IS NOT DISTINCT FROM gives false. Both reject the row in a WHERE or ON condition, so the change rewrites only conditions there, alone or joined by AND; under NOT or OR the or form stays.

Results

Benchmark in a greenfield project (H2), N records with 2 child records each, fastest of 3 runs in ms:

Case	N	Without the change	With the change

Temp-table inner join	1000 / 2000 / 4000	389 / 1533 / 6148	21 / 17 / 30
Temp-table OUTER-JOIN	1000 / 2000 / 4000	278 / 1149 / 4589	20 / 16 / 27
Database inner join	1000 / 2000 / 4000	389 / 1508 / 6122	19 / 24 / 42
Database OUTER-JOIN	1000 / 2000 / 4000	238 / 935 / 3772	18 / 21 / 36

Both builds returned the same rows in every case. A test of joins and filters on keys holding ? also prints the same output with and without the change. It matches the baseline except under NOT, where FWD misses the pairs with exactly one unknown key; that bug exists without the change too.

I created branch 12144a from trunk revision 16834 for this change. The fix is committed in rev. 16835.

#2 - 11/09/2026 07:00 AM - Stefanel Pezamosca

- Status changed from WIP to Review

#3 - 11/09/2026 07:08 AM - Stefanel Pezamosca

- Assignee set to Stefanel Pezamosca