Database - Bug #8459

investigate changing H2 to allow NULL to equal NULL (and the impact on the other compare operators)

03/15/2024 04:08 AM - Constantin Asofiei

Status:	WIP	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Radu Apetrii	**% Done:**	90%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

History

#1 - 03/15/2024 04:12 AM - Constantin Asofiei

- Assignee set to Radu Apetrii

- Status changed from New to WIP

With #8363, FWD now has a MANDATORY option allowed at temp-table fields. When using joins, this can have a huge impact, because the tt1.f1 IS NULL and tt2.f1 is null augmentation to address NULL equality (beside the tt1.f1 = tt1.f2 test) will make H2 use the scan index.

We need some answers here:

- how complicated is to make H2 allow NULL equality and the impact on the other compare operators
- Some preliminary investigation in using IS NOT DISTINCT FROM showed no improvement (or degradation). Is IS NOT DISTINCT FROM really not using the index? This is a double-check in case what I saw was incorrect.

Radu is investigating this now.

#3 - 03/15/2024 07:03 AM - Radu Apetrii

I created branch 8459a_h2 for this matter.

#4 - 03/15/2024 07:36 AM - Radu Apetrii

Constantin Asofiei wrote:

> will make H2 use the scan index.

Constantin, is there a chance that you can point me to where this choice of using the scan index is being made?
I made a slight adjustment to H2 and I want to make sure that things actually change.

#5 - 03/15/2024 07:38 AM - Alexandru Lungu

If you have two big tables and none have indexes, on join, it should do scan on both tables (nested join - N * M comparisons).

#6 - 03/15/2024 08:29 AM - Constantin Asofiei

This test shows the 'unknown equality' in OpenEdge:

```
def temp-table tt1 field f1 as int field f2 as int index ix1 f1.
def temp-table tt2 field g1 as int field g2 as int index ix2 g1.

create tt1.
tt1.f1 = 1.
tt1.f2 = 10.
release tt1.

create tt1.
tt1.f1 = ?.
tt1.f2 = 20.
release tt1.

create tt2.
tt2.g1 = 1.
tt2.g2 = 100.
release tt2.

create tt2.
tt2.g1 = ?.
tt2.g2 = 200.
release tt2.

for each tt1, each tt2 where tt1.f1 = tt2.g1:
   message tt1.f1 tt1.f2 tt2.g1 tt2.g2.
end.
```

In FWD, the join will be augmented as tt1.f1 = tt2.g1 or (tt1.f1 is null and tt2.g1 is null) - we want to remove the is null augmentation and change H2 so that 'null equals null'.

#7 - 03/18/2024 03:50 AM - Radu Apetrii

- File 8459.patch added

- % Done changed from 0 to 50

I've attached a patch for H2 in order to allow null equality. It certainly needs a review and a set of performance testing.
Also note that this change treats only the equality sign. If we decide to go forward with this approach, I'll allow null comparisons for the other symbols as well.
In terms of what was changed, I've removed a potential optimization (since we want to allow null comparisons), and changed the method that handles the equality between two components so that it permits null ones.

In the meantime, I'll work on the FWD side of changes and come back with updates shortly.

#8 - 03/18/2024 04:45 AM - Radu Apetrii

Alex/Constantin: Was the phrase tt1.f1 is null added just for the H2 dialect, or is this null equality an issue in multiple dialects? What I want to know is whether the removal of is null at FQL level would impact other dialects, or is it safe to handle it there. As an alternative, when converting the FQL into SQL, I can disable the addition of is null just when dealing with H2 dialect.

#9 - 03/18/2024 05:34 AM - Constantin Asofiei

Radu Apetrii wrote:

> Alex/Constantin: Was the phrase tt1.f1 is null added just for the H2 dialect, or is this null equality an issue in multiple dialects? What I want to know is whether the removal of is null at FQL level would impact other dialects, or is it safe to handle it there. As an alternative, when converting the FQL into SQL, I can disable the addition of is null just when dealing with H2 dialect.

The augmentation for is null is required for all dialects, not just H2. And if we make H2 to have null = null be true, then the augmentation in FQLPreprocessor needs to be skipped only for H2 dialect.

#10 - 03/18/2024 05:35 AM - Alexandru Lungu

Radu, I think we should restrict this with a flag. H2 is used with other goals in FWD and I don't what to make this change for every use-case.

Create a flag like in the H2_Database_Fork Wiki and use it to distinguish the cases where we actually need this. Add it in FWD for H2Helper connection string.

Please run the H2 tests and check which fail now - none should fail. Create a unit test in H2 for this flag (lets say USE_NULL_EQUALITY) and do extra tests.

As for the changes:

- I don't know how database.compare works, but I presume it will return false if only one parameter is null. I really want to avoid having a NPE somewhere after your changes. I would rather suggest to treat this **way** more specific, like if (useNullEquality && l null && r null) return true;, without letting the null values freely inside database.compare. It wasn't made to actually treat nulls, so I expect the worst.
- The same can go for getValue
- Please check the following construct ORDER BY x ASC NULLS LAST (how is the sorting done after your changes?). Also, check for indexes (CREATE INDEX idx ON tt (f1, f2 NULLS LAST)) and insert a record with f2 or f1 as null. Are they stored ok?

  > Alex/Constantin: Was the phrase tt1.f1 is null added just for the H2 dialect, or is this null equality an issue in multiple dialects? What I want to know is whether the removal of is null at FQL level would impact other dialects, or is it safe to handle it there. As an alternative, when converting

> the FQL into SQL, I can disable the addition of is null just when dealing with H2 dialect.

It is for all dialects. The feature you introduce should be done **only** for H2.

#11 - 03/19/2024 04:05 AM - Radu Apetrii

Alexandru Lungu wrote:

> Create a flag like in the H2_Database_Fork Wiki and use it to distinguish the cases where we actually need this. Add it in FWD for H2Helper connection string.

Should this setting be false by default? If so, then what should trigger it to become true?

#12 - 03/19/2024 04:09 AM - Constantin Asofiei

Radu Apetrii wrote:

> Alexandru Lungu wrote:
>
> > Create a flag like in the H2_Database_Fork Wiki and use it to distinguish the cases where we actually need this. Add it in FWD for H2Helper connection string.
>
> Should this setting be false by default? If so, then what should trigger it to become true?

Yes, default it to false; use a JDBC URL connection parameter to set it to true.

#13 - 03/19/2024 12:20 PM - Constantin Asofiei

Radu, do you have the FWD and H2 changes, for some early testing?

#14 - 03/19/2024 12:50 PM - Radu Apetrii

- Status changed from WIP to Review

- % Done changed from 50 to 100

Constantin Asofiei wrote:

> Radu, do you have the FWD and H2 changes, for some early testing?

Yes, sorry for the delay, I had quite a headache. The changes are on 8459a rev. 15064 and 8459a_h2 rev. 41.
I haven't finished testing, but so far results (in terms of correctness) look good.

Also, I'll be staying late tonight, so if the review and testing go well, I can help with anything else that is needed.

#15 - 03/19/2024 12:56 PM - Constantin Asofiei

Please rebase both 8459a_h2 and 8459a.

#16 - 03/19/2024 12:57 PM - Radu Apetrii

Constantin Asofiei wrote:

> Please rebase both 8459a_h2 and 8459a.

On it.

#17 - 03/19/2024 01:16 PM - Greg Shah

Who should be doing the review?

#18 - 03/19/2024 01:19 PM - Radu Apetrii

Radu Apetrii wrote:

> Constantin Asofiei wrote:
>
> > Please rebase both 8459a_h2 and 8459a.
>
> On it.

Done. 8459a rebased with trunk rev. 15069, and 8459a_h2 with fwd-trunk rev. 41.

> Who should be doing the review?

I'm OK with either Alex or Constantin. I'm not sure if Alex is still up, however.

#19 - 03/19/2024 03:11 PM - Alexandru Lungu

I am OK with the changes in 8459a (FWD branch).

For FWD-H2:

- In compare, you need to also explicitly handle the case when only one operand is null and return false
  - in other words, if the flag is set, one operand is null and the other is not, return false. With the current changes, it will return null.
  - the same reasoning goes with NOT_NULL

Please us the H2 console (java -jar fwd-h2-1.42.jar, or whatever your jar is named) and do a connection with jdbc:h2:mem:;use_null_equality=true;). Check your changes:

- select 1 = null
- select null = null
- select null = 1
- select 1 = 1
- the above with <>

The tests above should work normally without the flag.

#20 - 03/19/2024 04:03 PM - Radu Apetrii

Also, one more thing. In Comparison.getValue, depending on the compareType value, things should be treated for both = and != signs. I'm not sure if it is enough to test compareType 0 for equality and compareType 5 for non-equality. Aside from that, I have taken on board everything said.

#21 - 03/19/2024 06:58 PM - Radu Apetrii

I've committed to 8459a_h2 rev. 43 changes that include:

- Added a case for handling equality between a null component and a non-null component.
- Added TestNullEquality class responsible for null equality tests. This includes basic tests (like in #8459-19), tests with join (where both components are null), tests with ORDER BY tt.f1 NULLS LAST/FIRST, and tests with indexes that have NULLS LAST in them.

The only thing that I'm unsure about at the moment is the question raised in #8459-20. After this gets addressed, from my point of view, we will be in a good position.

#22 - 03/20/2024 03:24 PM - Alexandru Lungu

I am OK with 8459a_h2 changes. Good addition with the unit test!

The getValue looks right to me. The only concern is in regard to what 0 and 5 actually mean. Can we replace these with something more suggestive. Are these EQUAL and NON_EQUAL, or something like that?

#23 - 03/20/2024 04:54 PM - Radu Apetrii

Alexandru Lungu wrote:

> Can we replace these with something more suggestive. Are these EQUAL and NON_EQUAL, or something like that?

Yes, they represent EQUAL and NON_EQUAL. This was my concern as well, that maybe it is not the best practice to just check the equality with 0 or 5. Should I change these into something else?

#24 - 03/21/2024 03:35 AM - Alexandru Lungu

What about simple comparison with EQUAL (instead of 0) and NON_EQUAL (instead of 5)? Am I missing something?

#25 - 03/21/2024 03:43 AM - Radu Apetrii

Alexandru Lungu wrote:

> What about simple comparison with EQUAL (instead of 0) and NON_EQUAL (instead of 5)? Am I missing something?

No, I was a bit confused, because I thought these were values for a bitmask. I didn't look properly.

I replaced the values 0 and 5 with EQUAL and NOT_EQUAL in 8459a_h2 rev. 44.

#26 - 03/21/2024 03:48 AM - Alexandru Lungu

- Status changed from Review to Internal Test

The unit tests in H2 suffice. I think you are good to go with the merge of 8459a_h2.

Constantin, anything to add here?

Otherwise, lets follow: merge H2 changes, build FWD-H2, upload the latest FWd-H2 to the public repository, update the version in 8459a (build.gradle) to pick up the new binary, **test**, prepare merge for 8459a.

#27 - 03/21/2024 05:20 AM - Constantin Asofiei

Alexandru, I'm good.

Radu: if unit tests/standalone tests are OK, please build fwd-h2 and upload it to devsrv01:/tmp/. I'll upload it to the public repository. Once this and 8459a is ready, I'll do ETF testing.

#28 - 03/21/2024 05:42 AM - Radu Apetrii

Constantin Asofiei wrote:

> Alexandru, I'm good.
>
> Radu: if unit tests/standalone tests are OK, please build fwd-h2 and upload it to devsrv01:/tmp/. I'll upload it to the public repository. Once this and 8459a is ready, I'll do ETF testing.

Done. I've uploaded the jar to devsrv01:/tmp/. The name is fwd-h2-1.42-trunk.jar.

#29 - 03/21/2024 06:29 AM - Constantin Asofiei

fwd-h2 1.42 is on projsrv01.

#30 - 03/21/2024 02:04 PM - Constantin Asofiei

I've started merge of 8459a.

#31 - 03/21/2024 02:12 PM - Constantin Asofiei

- Status changed from Internal Test to Test

Branch 8459a was merged into trunk as rev. 15082 and archived.

#32 - 03/22/2024 03:52 AM - Constantin Asofiei

- Status changed from Test to WIP

- Priority changed from Normal to Immediate

There is a regression in #7143-784 related to the NULL changes in H2. If I comment use_null_equality=true;, then the scenario works.

To replicate use this:

```
def temp-table tt1 field f1 as int field f2 as char index ix1 is unique primary f1.

def var i as int.
do i = 1 to 10 transaction:
   create tt1.
   assign tt1.f1 = (if i mod 2 = 0 then ? else i) tt1.f2 = string(i).
   release tt1.
end.

for each tt1: message tt1.f1 tt1.f2. end.
```

Alexandru: I'm inclined to disable use_null_equality=true; until this is fixed. The problem is this happens only when we do a unique validation using SELECT, and not on INSERT - if you remove the TRANSACTION, then it works.

#33 - 03/22/2024 03:55 AM - Constantin Asofiei

```
def temp-table tt1 field f1 as int index ix1 is unique primary f1.

def var i as int.
do transaction:
   create tt1.
   tt1.f1 = ?.
   release tt1.
   create tt1.
   tt1.f1 = ?.
   release tt1.
end.

for each tt1: message tt1.f1. end.
```

#34 - 03/22/2024 04:04 AM - Alexandru Lungu

Created 8459b and committed the disable for USE_NULL_EQUALITY.

Preparing for merge with trunk.

#35 - 03/22/2024 04:10 AM - Alexandru Lungu

Merged to trunk as rev. 15085.

#36 - 03/22/2024 04:16 AM - Radu Apetrii

Constantin Asofiei wrote:

> The problem is this happens only when we do a unique validation using SELECT, and not on INSERT - if you remove the TRANSACTION, then it works.

This means that the error appears when performing checks at the end transaction level, right? (and not inside the transaction, even though release s happen)

#37 - 03/22/2024 04:22 AM - Constantin Asofiei

Alexandru Lungu wrote:

> Merged to trunk as rev. 15085.

Please post to #7847 also.

#38 - 03/22/2024 05:07 AM - Alexandru Lungu

Radu Apetrii wrote:

> Constantin Asofiei wrote:
>
> > The problem is this happens only when we do a unique validation using SELECT, and not on INSERT - if you remove the TRANSACTION, then it works.
>
> This means that the error appears when performing checks at the end transaction level, right? (and not inside the transaction, even though release s happen)

I think it is the release that is causing this. RELEASE is forcefully calling the WRITE triggers according to the documentation, so it is usually associated with flush and validation. Also, I think DO TRANSACTION is redundant; you would have the same problem if you attempted to do tt1.f1 = ? the second time.

Radu, mind that you can also use VALIDATE statement to trigger validation.

#39 - 03/22/2024 01:36 PM - Radu Apetrii

- % Done changed from 100 to 90

Well, this is pretty bad in my opinion. I don't really have a solution for this, I need to think this through a bit more.

It seems unnatural to me that Progress allows duplicate records that have ?, even though an unique index has been defined on that column. The most logical reason that I was able to find is: "Since null is an undefined state, Progress doesn't see it as a value, so it can't be seen as duplicate."

Before the null equality changes, when the index validation would happen, the program would return null instead of true, so the duplicate condition wouldn't be satisfied, thus the error wouldn't appear. Any suggestions are happily accepted.

#40 - 03/25/2024 02:53 AM - Alexandru Lungu

- Priority changed from Immediate to Normal

There should be a distinction between equality check when updating/inserting and the equality check when querying. We should honor the flag only when selecting. **Or** disable the flag when inserting/updating.

In FWD-H2, there are usually USE statements to enable/disable a feature. The alternative is to use direct-access like the validateMode.

I think this task can be reduced in priority. We need to have this done to improve performance in general, but I think there is no longer an urgency for it (right Constantin?).

Files

8459.patch	980 Bytes	03/18/2024	Radu Apetrii