Database - Feature #7194

Avoid generating an ORDER-BY clause if not required

03/14/2023 09:49 AM - Alexandru Lungu

Status:	Test	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Radu Apetrii	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**version:**	
vendor_id:	GCD		

Description

History

#1 - 03/14/2023 10:02 AM - Alexandru Lungu

- Tracker changed from Bug to Feature

This task targets the following scenario:

```
define temp-table tt field f1 as int index idx1 as unique primary f1.
create tt.
tt.f1 = "abc".
find first tt where tt.f1 = "abc".
```

Which is converted to:

```
tt.create();
tt.setF1(new character("abc"));
new FindQuery(tt, "upper(tt.f1) = 'ABC'", null, "tt.f1 asc").first();
```

This kind of scenario can be extended to CAN-FIND. The issue I encountered is that the final query SQL looks like:

```
select [...] where upper(rtrim(tt.f1)) = 'ABC' order by upper(rtrim(tt.f1)) asc limit 1
```

This is expected, as we want to retrieve the first tt which matches the where clause and the provided order-by clause. However, we can detect that this query is going to have at most one result due to FQLPreprocessor.isUniqueFind (note that f1 is uniquely indexed). Therefore, there is no need for the order by and limit clauses, allowing more candidate plans to the database engine and reducing the parsing time.

After detecting that such query is a "unique" find, I expect the following SQL to work similarly:

```
select [...] where upper(rtrim(tt.f1)) = 'ABC'
```

I am aware the there are no database level unique indexes and that FWD manages the database unique constraint at the server level. However, I am not sure how this change will interact with this. Needs further investigation.

#2 - 03/28/2023 09:23 AM - Alexandru Lungu

- Assignee set to Radu Apetrii

Radu, please address #7194 once you get a stable solution for #7066. AFAIK, you already have some implementation regarding the suppression of the ORDER BY clause, as part of the introduction of USE-INDEX. Try to make use of that to cover these cases of "unique index look-up" where ORDER BY is not needed.

#3 - 03/29/2023 06:37 AM - Eric Faulhaber

Is the idea to just remove the ORDER BY at the SQL level (as opposed to the FQL level)?

If you are intending to remove it at the FQL level, please note that we do a reverse lookup on the ORDER BY clause early in RandomAccessQuery processing, and there is some RecordNursery processing that is dependent upon that. There may be other runtime dependencies on knowing the legacy index in use as well. Please keep these in mind, such that we don't introduce regressions with this optimization.

#4 - 03/29/2023 07:17 AM - Alexandru Lungu

Eric Faulhaber wrote:

> If you are intending to remove it at the FQL level, please note that we do a reverse lookup on the ORDER BY clause early in RandomAccessQuery processing, and there is some RecordNursery processing that is dependent upon that. There may be other runtime dependencies on knowing the legacy index in use as well. Please keep these in mind, such that we don't introduce regressions with this optimization.

I was thinking of SQL. Radu, please advice; are your changes in #7066 at SQL level?

#5 - 03/30/2023 05:36 AM - Radu Apetrii

Alexandru Lungu wrote:

> Eric Faulhaber wrote:
>
> > If you are intending to remove it at the FQL level, please note that we do a reverse lookup on the ORDER BY clause early in RandomAccessQuery processing, and there is some RecordNursery processing that is dependent upon that. There may be other runtime dependencies on knowing the legacy index in use as well. Please keep these in mind, such that we don't introduce regressions with this optimization.
>
> I was thinking of SQL. Radu, please advice; are your changes in #7066 at SQL level?

My changes are at FQL level.

I will write a more detailed post in #7066, but, long story short, the ORDER BY is removed from the FQL only **when dealing with a multi-table non-invalidated (PreselectMode) AdaptiveQuery over temporary tables**. In the other cases, it is exactly the same. I thought that by removing the ORDER BY from the FQL, then that component would not require to be parsed in order to convert it to SQL (so a small performance boost would be obtained).

On its way to being converted to SQL, is the FQL (of the previously described case) required in other parts of the program? If so, I can adapt the

solution.

#6 - 04/06/2023 07:37 AM - Radu Apetrii

I've added in PreselectQuery a function that checks if the query will retrieve a unique finding. If so, then no ORDER BY clause should be required.
The changes are on 7026c, rev. 14527.

After navigating a large customer application for 10-15 minutes, the results showed ~40 queries that successfully decided to not include the ORDER BY clause in their FQL.
Although the changes are at FQL level, as far as I've tested, things don't interfere with RandomAccessQuery or RecordNursery, but I will wait for a review.

#7 - 04/07/2023 08:39 AM - Alexandru Lungu

- Status changed from New to WIP

- % Done changed from 0 to 80

#8 - 04/24/2023 05:34 AM - Alexandru Lungu

Radu, let the intended changes for #7194 reside in 7026c. I am thinking of merging 7026c independently from 7066a, as 7066a needs more testing. Please let me know if you have changes in 7066a that should be moved to 7026c first. Afterwards, I will start reviewing.

#9 - 04/25/2023 10:34 AM - Alexandru Lungu

My initial thought when I created #7194 was to optimize UNIQUE FIND/CAN-FIND for persistent databases. For temp databases we already have direct access which doesn't take the order-by into account. My concern was to avoid sending an ORDER BY to PostgreSQL, for example, even though we know that the UNIQUE FIND/CAN-FIND is going to find at most 1 record.

However, it is interesting that you tackled the PreselectQuery this way and that you actually find unique where clauses. At a certain extent I think we can even resolve unique PreselectQuery to direct-access (for _temp), but I should put some thought on this.

Review of 7026c, rev. 14527
This change conditionally removes the ORDER BY from PreselectQuery and AdaptiveQuery if not needed (when each component is marked as isUniqueFind).

- The changes are clean enough.
- My regression testing passes.
- Even if this doesn't regress, I am not 100% certain that removing the ORDER BY will not break the AdaptiveQuery.
  - What if the uniquely found record is changed? The query should invalidate (or not?) and "refind" the unique record again in dynamic mode. Radu, please make an example with this - I hope at this point th invalidation will trigger so that ORDER BY won't matter anyway.
  - I don't expect the LAZY mode to interact with your ORDER BY in any way. Please double-check this.
  - Is the adaptive_scrolling/non-scrolling test-suite OK with your changes? For both persistent (concurrent environment) and _temp (sometimes using LAZY)?
- Profiling shows the following:

Scenario	Count	Time (ms)	Count %	Time %
ORDER BY required	28,194	85.033	79.7%	82.9%
ORDER BY not required	7,164	17.465	20.3%	17.1%

This is the time to actually append the ORDER BY clause into the FQL; it is not related to SQL execution time. With the changes, I expect to see 20% less ORDER BY generations (faster with ~17ms) and fortunately faster SQL parsing, planning, executions, caching, SQL conversion, etc. I will run my profiling now.

#10 - 04/26/2023 10:18 AM - Alexandru Lungu

- Status changed from WIP to Review

- % Done changed from 80 to 100

Tested this. I got a -0.24% performance improvement with a 7026c profile run.
Just rebased 7026c with trunk.

Fully reviewed tested and profiled. 7026c is ready for trunk.
I intent to bump the last FWD-H2 revision with 7026c (rev. 17).

#11 - 04/26/2023 10:21 AM - Alexandru Lungu

- Status changed from Review to WIP

- % Done changed from 100 to 50

I will move this back to 50% to allow the handling of other ORDER BY clauses (from RandomAccessQuery especially).

#12 - 04/28/2023 04:12 AM - Radu Apetrii

Alexandru Lungu wrote:

- What if the uniquely found record is changed? The query should invalidate (or not?) and "refind" the unique record again in dynamic mode. Radu, please make an example with this - I hope at this point the invalidation will trigger so that ORDER BY won't matter anyway.

From the tests I've done, the program works as before, regarding the invalidation process. This is quite a tricky test though, having a query that retrieves at most one record, which should/can/might be "re-found" later. It's not like after a bunch of next s the record can be found again. By this I mean that the test is very specific and from my gatherings it works fine.

- Is the adaptive_scrolling/non-scrolling test-suite OK with your changes? For both persistent (concurrent environment) and _temp (sometimes using LAZY)?

Yes, the results were as expected for all the tests (both persistent and temporary). However, very few unique findings were encountered in the test-suite.

In #7194-5 I said that this feature is available only in a multi-temporary-table environment. This is not true. I think my ideas got mixed up when writing the post for #7066, sorry. **At the moment of speaking, this is available for all PreselectQueries and AdaptiveQueries, regardless of persistent/temporary and single/multi-table environments.**

- I don't expect the LAZY mode to interact with your ORDER BY in any way. Please double-check this.

Taking into account the previous point, LAZY can appear when dealing with a single-table query (which has a unique finding) over a temporary table. I don't think that this should be a problem, especially considering the fact that some tests included this exact case. If this happens to be wrong, I will adapt the solution accordingly.

#13 - 05/09/2023 04:01 AM - Radu Apetrii

I have modified RandomAccessQuery so that an ORDER BY clause will not be added to the FQL if a unique finding is encountered. This takes place when creating a bundle (in FQLHelper.createBundle), only in FIRST/LAST cases and only for persistent databases.
The changes are on 7026d, rev. 14562.
I've tested the changes with some examples and a large customer application. In the last case, ~28.74% of the FIRST/LAST RandomAccessQueries over persistent databases lead to a unique finding, thus orderBy became null.

#14 - 05/25/2023 08:34 AM - Alexandru Lungu

The changes are tested and OK to me.

However, I can't isolate a performance increase for this change (it is either the same as before or slightly worse (+0.1%) on some profiling runs. Can you profile some dummy testcases with/without the change? Maybe that order-by was helping the planning as a hint .... or not. (only RAQ changes).

#15 - 05/26/2023 06:24 AM - Radu Apetrii

Alexandru Lungu wrote:

> Can you profile some dummy testcases with/without the change?

To be honest, the difference is quite unnoticeable. I tried a test with/without the changes and:

- If 10,000 queries or less were executed, one could not distinguish a difference between them.
- For 100,000 queries, a -1.9% improvement could be seen (10.4 seconds without the changes vs 10.2 seconds with the changes).

Some additional notes:

- FQLHelperCache was disabled for this round of testing which leads me to believe that in real scenarios, this improvement percentage is even smaller.
- All queries that were executed lead to a unique finding.
- That -1.9% improvement refers only to the total time of executing the queries, not the program itself.

#16 - 05/29/2023 03:03 AM - Alexandru Lungu

- % Done changed from 50 to 80

There is an idea of a test-case that troubled my mind for a while :)

Can you check if we can make FQLPreprocessor.isUniqueFind vulnerable? AFAIK, inside a DO TRANSACTION block we can have two records (in two different buffers) sharing the same value for a field even if the field is unique. The check is done at the end of DO TRANSACTION. Until that point, we can violate the constraint as long as we leave the transaction is a correct state. This example is somewhere latent in my head and can be wrong.

Try to make a two other buffers in the transaction to use with FIND FIRST and FIND LAST and a WHERE on the uniquely indexed field. Do these retrieve two different records? Try with _temp and persistent on 4GL and FWD, with your changes. If the tests works and provide different results in 4GL, I think we have a problem. Note that _temp tests are needed even to check direct access.

Note: temp database doesn't have unique indexes; the constraints are checked in FWD before each flush. The persistent database has unique indexes; any violation will throw an SQLException and handled properly in FWD.

#17 - 05/31/2023 02:47 AM - Alexandru Lungu

Radu, please post your attempt on #7194-16 here (the test-case). AFAIK, you couldn't generate two records with the same value violating a unique constraint for the period of the transaction.

#18 - 05/31/2023 07:26 AM - Radu Apetrii

Alexandru Lungu wrote:

> Radu, please post your attempt on #7194-16 here (the test-case). AFAIK, you couldn't generate two records with the same value violating a unique constraint for the period of the transaction.

Right. I tried to run the following test in 4GL, but I received an error saying b2 already exists with f1 1.

```
define temp-table tt field f1 as int field f2 as int index idx1 as unique primary f1.
define buffer b1 for tt.
define buffer b2 for tt.

do transaction:
    create b1. b1.f1 = 1. b1.f2 = 1.
end.

do transaction:
    create b2. b2.f1 = 1. b2.f2 = 2.

    define buffer b3 for tt.
    define buffer b4 for tt.

    find first b3 where b3.f1 = 1.
    find last b4 where b4.f1 = 1.

    message b3.f1 b3.f2 b4.f1 b4.f2.
    b2.f1 = 2.
end.
```

Note: I also tried to create the two records in the same transaction, but the error remains the same.

If there are any suggestions, I will be thankful.

#19 - 06/30/2023 02:49 AM - Alexandru Lungu

- Status changed from WIP to Review

- % Done changed from 80 to 100

Radu, can you try creating the records in different transactions. I don't have a clear perspective yet on how the test should look like, but I think you can get a lot o feedback from #7323 (unique checks). The goal here is to force FWD-H2 to have two records which don't satisfy the unique constraint in the database in the same time. I hope this is not possible, but I need a more solid confirmation on that.

I will move this to review/100% just because everything is done already - this last step are only some correctness checks.

#20 - 07/21/2023 10:42 AM - Eric Faulhaber

Alexandru Lungu wrote:

> I will move this to review/100% just because everything is done already - this last step are only some correctness checks.

What exactly does this mean? AFIACT, the changes have been integrated into FWD-H2, correct? So, what is the actual status of this task?

#21 - 07/21/2023 10:51 AM - Alexandru Lungu

- Status changed from Review to Test

Eric Faulhaber wrote:

> What exactly does this mean? AFIACT, the changes have been integrated into FWD-H2, correct? So, what is the actual status of this task?

It is mostly about a presumption we made initially that we wanted to test 100%. In the mean-time, there were some other issues that popped out, not necessarily due to the changes here: #7535, #7474. This can be closed; other side-issues will be discussed in the other tasks.