Database - Bug #7174

Resolve simple CAN-FIND statements faster

03/07/2023 04:40 AM - Alexandru Lungu

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

Description

Related issues:

Related to Database - Feature #7076: make CAN-FIND avoid hydration	**Closed**
Related to Database - Bug #7185: H2 in-memory lazy hydration	**Test**

History

#1 - 03/07/2023 04:59 AM - Alexandru Lungu

- Assignee set to Dănuț Filimon

- File empty_can_find.patch added

- Status changed from New to WIP

There are some opportunities to short-circuit the CAN-FIND query in FWD. A CAN-FIND statement is relaxed, in the sense that it shouldn't update the buffer, throw persistence errors or lock records. Please read the CAN-FIND documentation before moving on with this to ensure that short-circuiting doesn't skip any important step in the way.

Right now, a CAN-FIND is resolved through a RandomAccessQuery. It either uses the ffCache or executes an SQL query. Note that CAN-FIND can state a where clause and can be ANY (does the table have at least one record?) or ONE (does the table have exactly one record?). Please use 7026a for now, to have access to the latest changes regarding direct-access.

Most of the optimizations are related to temporary tables:

- Simply return false if the buffer is backed by a definitely empty table. This is already handled in the provided patch.
- Simply return true if the buffer is backed by a definitely non-empty table, it doesn't have a where clause and is ANY.
- Use the FQLPreprocessor to detect if this is a findByRowId or findByUniqueIndex query. If so, use ANY instead of ONE. I guess ANY is wider and, thus, faster. Check if this is the case.
- Consider extending direct-access technique (RandomAccessQuery.executeDirectAccess) to cover some specific use-cases.

I am still in the process of debugging some large-customer applications to make a statistic of how many simple queries are executed and how they look like.

#2 - 03/07/2023 05:08 AM - Igor Skornyakov

Please note that we have a task regarding CAN-FIND (#7113) waiting for the code review.
Maybe it makes sense to wait until the corresponding changes will be merged to the trunk?

#3 - 03/14/2023 05:29 AM - Dănuț Filimon

I created tests that use simple queries, those tests consist of executing CAN-FIND for each record in a temporary/persistent table. I compared CanFind.ONE and CanFind.ANY on tables with 10k records and got the following results:

Temp-table	CanFind.ONE (ms)	CanFind.ANY (ms)	Difference
no index	9009.6	4530.2	-49.718%
with index	151.2	131.2	-13.227%

Persistent table	CanFind.ONE (ms)	CanFind.ANY (ms)	Difference
no index	10485.4	8688.8	-17.134%
with index	10398.6	5755.6	-44.650%

Alexandru Lungu wrote:

> Most of the optimizations are related to temporary tables:
>
> - Simply return false if the buffer is backed by a definitely empty table. This is already handled in the provided patch.
> - Simply return true if the buffer is backed by a definitely non-empty table, it doesn't have a where clause and is ANY.
> - Use the FQLPreprocessor to detect if this is a findByRowId or findByUniqueIndex query. If so, use ANY instead of ONE. I guess ANY is wider and, thus, faster. Check if this is the case.

ANY can also be used instead of ONE for persistent tables.

After including the three mentioned optimizations and testing again, I documented the results and compared them to the **unpatched version of 7026b**:

Temp-table	CanFind.ONE - No patch (ms)	CanFind.ONE - Patched (ms)	Difference (1)	CanFind.ANY - No patch (ms)	CanFind.ANY - Patched (ms)	Difference (2)
no index	9009.6	8661.8	-3.860%	4530.2	4662	+2.909%
with index	151.2	140	-7.407%	131.2	146	+11.280%

Persistent table	CanFind.ONE - No patch (ms)	CanFind.ONE - Patched (ms)	Difference (1)	CanFind.ANY - No patch (ms)	CanFind.ANY - Patched (ms)	Difference (2)
no index	10485.4	10630.8	+1.386%	8688.8	9073.4	+4.426%
with index	10398.6	5874.2	-43.509%	5755.6	5903.4	+2.567%

There is an improvement when using ANY instead of ONE for persistent tables when a findByUniqueIndex is detected. The rest of the results don't have a high margin, except the +11% when using ANY for temp-tables.

Committed 7026b/rev.14502. Added short-circuits for CAN-FIND.

#4 - 03/14/2023 07:12 AM - Alexandru Lungu

- Related to Feature #7076: make CAN-FIND avoid hydration added

#5 - 03/14/2023 07:14 AM - Alexandru Lungu

- Related to Bug #7185: H2 in-memory lazy hydration added

#6 - 03/14/2023 09:23 AM - Greg Shah

In OE, will a CAN-FIND ONE result return true if there is more than one match? In other words, is CAN-FIND ONE meant to answer the question: "Does this record exist as a unique result?".

#7 - 03/14/2023 09:35 AM - Alexandru Lungu

These are the expected results, where ONE is CAN-FIND(tt), ANY is CAN-FIND(FIRST tt).

Number of records	ONE	ANY
0	no	no
1	yes	yes
> 1	no	yes

Short answer, yes: CAN-FIND ONE is meant to answer the question: "Does this record exist as a unique result?".

#8 - 03/14/2023 12:30 PM - Eric Faulhaber

This semantic is why we have the has{Any|One} nomenclature in FindQuery.

- CAN-FIND({FIRST|LAST} ...) is converted to:
  - FindQuery.hasAny(...) (i.e., we don't care how about uniqueness, just tell if any record meeting the criteria exists) in the standalone case
  - a subselect using exists(...) in the nested in a WHERE clause case
- CAN-FIND(...) (no special keyword; uniqueness is implied) is converted to:
  - FindQuery.hasOne(...) (i.e., tell if there is one (and only one) instance of the record meeting the criteria) in the standalone case
  - a subselect using count(select <primary key> where ...) = 1 in the nested in a WHERE clause case

#9 - 03/15/2023 05:25 AM - Alexandru Lungu

- Status changed from WIP to Review

- % Done changed from 0 to 100

There is an improvement when using ANY instead of ONE for persistent tables when a findByUniqueIndex is detected. The rest of the results don't have a high margin, except the +11% when using ANY for temp-tables.

- The first optimization for empty tables is not caught by your tests - you mention "10k records" per table.
- I don't know if the second optimization is hit; are you testing with no where clauses to match the no constraint precondition?
- The latest optimization converts ONE to ANY where possible. In your tests, ANY is always faster than ONE, so it is a good optimization overall.

I think your <10% rates are error thresholds, considering that the first two optimizations are not hit. Correct me if I am wrong.

Review of 7026b/rev. 14502

- Danut, please update the copyright each type you are modifying a file (i.e. RandomAccessQuery). Also, update the # in history entry if the changes come from another branch then the previous entry. In your case, this is the first entry on RandomAccessQuery from 7026b.
- Move canFindMode = = CanFind.ANY to be the first clause in your guard. It is the fastest and has a good variance comparing to the other clauses (buffer.isTemporary() && !hasConstraints()).
- buffer.isTemporary is a bit confusing as it is not equivalent to our intent: "the table is not empty" (even if conceptually isTableDefinitelyEmpty was already used). Please create a definitelyHasRecords. Make it return false for RecordBuffer and !isTableDefinitelyEmpty() for TemporaryBuffer. This way, we can replace buffer.isTemporary to buffer.definitelyHasRecords/
- It is best to add a canFindMode == CanFind.ONE to the helper.getFQLPreprocessor().isUniqueFind(true) guard. I see a lot of hits here updating canFindMode to ANY, even if it was already set on ANY.

Tested 7026b/rev. 14503 and got -1.8% improvement overall.
There was no regression in the POCs I tested.

#10 - 03/16/2023 08:38 AM - Dănuț Filimon

Alexandru Lungu wrote:

> I think your <10% rates are error thresholds, considering that the first two optimizations are not hit. Correct me if I am wrong.

The only important part of the results is that ANY is always faster than ONE and that it shows an improvement when using persistent tables, not just temporary tables. The rest of the results are error thresholds as you said.

Committed 7026b/rev.14508. I applied the changes based on the mentioned points.

#11 - 04/20/2023 07:14 AM - Alexandru Lungu

- Status changed from Review to Test

This was merged in trunk as rev. 14523 and can be closed.

Files

empty_can_find.patch	662 Bytes	03/07/2023	Alexandru Lungu