Database - Feature #6928

H2 UPDATE performance improvement

11/15/2022 03:49 AM - Constantin Asofiei

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Constantin Asofiei	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**version:**	
vendor_id:	GCD		

Description

Related issues:

Related to Database - Support #7058: get the FWD fork of the H2 code base und...	**WIP**

History

#1 - 11/15/2022 06:45 AM - Constantin Asofiei

- File h2_update_perf.patch added

In H2, when doing an UPDATE statement,the approach is to:

- create a new row
- assign all columns not part of the UPDATE to the old version of the row
- re-calculate computed columns
- assign the columns in the UPDATE statement to the new row
- remove the old row from all indexes
- add the new row to all indexes

This approach doesn't look optimal:

- in tables with 100s of columns, all these columns are iterated, even in cases when only one column is included in the update
- if the UPDATE doesn't change indexed columns, then the indexes are updated, regardless if it doesn't affect them. This index re-calculation is expensive (as it wants to rebalance the tree, etc).

The attached patch is a first attempt to improve the UPDATE statement by not going through all the table columns, and 'stay' only on the ones targeted by the UPDATE statement. It needs further refinement and ensuring that it doesn't break the existing code.

A way to build a 4GL test to stress-test this is to:

- have a table with 50-100 fields, a few of them part of indexes
- create 10k records
- do some 10k assignments and test indexed and non-indexed fields.

The 4GL program can look like this:

```
def temp-table tt1 field f1 as int field f2 as int field f3 as int index ix1 f1.
def var i as int.

do i = 1 to 100:
 create tt1.
 tt1.f1 = i * 100 + 1.
 tt1.f2 = i * 100 + 2.
 tt1.f3 = i * 100 + 3.
 release tt1.
end.

find first tt1 where tt1.f1 = 50 * 100 + 1.

// stress-test indexed update
do i = 1 to 100:
   tt1.f1 = - tt1.f1.
```

```
end.

// stress-test non-indexed update
do i = 1 to 100:
   tt1.f2 = - tt1.f2.
end.
```

What's important is that when tt1.f1 or tt1.f2 is being assigned, FWD will go and 'validateMaybeFlush' the change immediately, so H2 executes an UPDATE statement.

#2 - 11/28/2022 09:51 AM - Constantin Asofiei

- Status changed from New to WIP

- Assignee set to Constantin Asofiei

#3 - 11/28/2022 10:18 AM - Constantin Asofiei

- File updateperf.p added

- File h2-1.4.200.jar added

- File h2_update_perf2.patch added

Attached is the second version of the patch and a stress-test for the UPDATE statement (10k field setters). The 'in-place' update can be done only if:

- the database is in memory and page-store
- only non-indexed columns are affected.

This is required because H2 keeps a single instance of the row in all indexes; and when an index needs to be updated, this can be done only by removing and re-adding the row. It will not work to update only the affected indexes, as all indexes need to share the same row instance. Another approach would be to 'update in place' regardless if an index is affected, and then recalculate only the affected indexes; but this may be more time-consuming than just remove/add the row.

The results are (5 runs, indexed/non-indexed update times, in millis). In OE, the times is 20/3 for indexed/non-indexed cases.

- with fwd-h2-1.4.200-6.jar version:

```
1: 204/116
2: 92/95
3: 88/87
4: 89/88
5: 88/86
```

- with the attached patch:

```
1: 83/57
2: 55/26
3: 59/23
4: 60/27
5: 55/23
```

Alexandru/Ovidiu: I need this tested on some other machine. Please run the attached test, with the attached h2-1.4.200.jar and with fwd-h2-1.4.200-6.jar file, 5 times, and post the results. Use the ChUI client.

In the mean time, I'll do some automated testing.

I've been running the H2 unit tests, too, and this showed some problems which I had to fix, but the unit tests with rev 7 of the fwd-h2 project still have some failures. I can't tell if these are from our changes or not.

#4 - 11/28/2022 03:15 PM - Ovidiu Maxiniuc

With attached h2-1.4.200.jar, 1st run:

n	indexed	non-indexed
1	799 ms	453 ms
2	302 ms	270 ms
3	202 ms	276 ms
4	200 ms	216 ms
5	314 ms	237 ms
Avg	254.50 ms	249.75 ms
%	101.90	

Second run (restarted the server):

n	indexed	non-indexed
1	596 ms	311 ms
2	285 ms	268 ms
3	328 ms	251 ms
4	228 ms	275 ms
5	254 ms	258 ms
Avg	273.75 ms	263.00 ms
%	104.09	

With old fwd-h2-1.4.200-6.jar:

n	indexed	non-indexed
1	570 ms	368 ms
2	201 ms	206 ms
3	194 ms	238 ms
4	238 ms	210 ms
5	195 ms	206 ms
Avg	207.00 ms	215.00 ms
%	96.28	

Second run (restarted the server):

n	indexed	non-indexed
1	855 ms	497 ms
2	257 ms	248 ms
3	248 ms	246 ms
4	235 ms	237 ms
5	216 ms	210 ms
Avg	239.00 ms	235.25 ms
%	101.59	

Constantin, notice that I run the tests twice for each jar. For several reasons:

- my times were generally 3-4x times greater than yours;
- I did not observe much improvement with the new jar. In fact the old library seems to be marginally faster. So I delete/copy the files again, just to be sure I did not messed the jars;
- the initial run showed the indexed case performing slower (+2%) than non-indexed with the new jar, while with the old one the indexed was 4% faster. In the second run both non-indexed updates performed better (?), by +4% and +2% respectively. That's strange! (Note: when doing the statistics I used only the client executions 2-5, ignoring the 1st, cold run).

#5 - 11/29/2022 04:16 AM - Constantin Asofiei

Thanks, Ovidiu, I'll profile some more and see what happens.

#6 - 11/29/2022 05:34 AM - Alexandru Lungu

I am doing the stress tests right now.

> I've been running the H2 unit tests, too, and this showed some problems which I had to fix, but the unit tests with rev 7 of the fwd-h2 project still have some failures. I can't tell if these are from our changes or not.

When I introduced the 7th revision, some unit tests failed. In some cases, the tests expected a scan index to be used instead of a tree index. My changes improved the query plan to use tree index in some certain cases which proved faster. You can safely fix the unit tests. I guess I forgot to do the unit test changes in rev. 7.

#7 - 11/29/2022 05:59 AM - Alexandru Lungu

version	server run	client run	indexed (ms)	unindexed(ms)
fwd-h2-1.4.200-6.jar	1	1	270	181
fwd-h2-1.4.200-6.jar	1	2	144	142
fwd-h2-1.4.200-6.jar	1	3	141	142
fwd-h2-1.4.200-6.jar	1	4	142	149
fwd-h2-1.4.200-6.jar	1	5	137	136
Warm avg.			**141**	**142**
fwd-h2-1.4.200-6.jar	2	1	369	188
fwd-h2-1.4.200-6.jar	2	2	154	148
fwd-h2-1.4.200-6.jar	2	3	141	149
fwd-h2-1.4.200-6.jar	2	4	135	139
fwd-h2-1.4.200-6.jar	2	5	137	137
Warm avg.			**142**	**143**
version	server run	client run	indexed (ms)	unindexed(ms)
h2-1.4.200.jar	1	1	349	178
h2-1.4.200.jar	1	2	156	160
h2-1.4.200.jar	1	3	145	152
h2-1.4.200.jar	1	4	165	195
h2-1.4.200.jar	1	5	138	143
Warm avg.			**151**	**160**
h2-1.4.200.jar	2	1	353	209
h2-1.4.200.jar	2	2	153	153
h2-1.4.200.jar	2	3	143	178
h2-1.4.200.jar	2	4	137	140
h2-1.4.200.jar	2	5	136	146
Warm avg.			**142**	**154**

Note that there is a recent addition to Hotel GUI regarding H2 performance testing. Pressing CTRL-ALT-H anywhere pops up a H2 profiling dialog. There are only some tests which compute their etime in a browse (for create, update, insert and delete). This was designed explicitly for profiling H2 version in different cases. However, I guess this UPDATE test is OK to be shared here as it requires a fine profiling (in terms of ms / in ChUI). Anyways, that tool can be upgraded to cover many more scenarios (like this one).

#8 - 12/02/2022 02:32 PM - Constantin Asofiei

- File h2-1.4.200.jar added

- File h2_update_perf3.patch added

Alexandru/Ovidiu: previously I uploaded the wrong jar (it was the rev 7 one, not the patched one). Please try with the attached.

Alexandru: I recall there were some performance testcases written for H2, can you point me where these are?

#9 - 12/02/2022 04:52 PM - Ovidiu Maxiniuc

Here are the results for two server runs, each with 5 client connection:

First Run				Second Run		
No	indexed (ms)	non-indexed (ms)		No	indexed (ms)	non-indexed (ms)
Cold	356	188		Cold	326	138
1	136	68		1	125	52
2	109	62		2	160	132
3	142	74		3	124	57
4	115	64		4	122	105
Avg	**125.5**	**67**		**Avg**	**132.75**	**86.5**
%	+187.3			%	+153.5	

It actually have improved a **lot**, an odd issue which can be visible is that the indexed executions take up to twice as much time. However, as strange as this looks, it is actually **normal**: since there are no indices to update, the non-indexed variant is faster! It is the sacrifice which needs to be paid to get a better indexed access later. Taking into balance that updates usually happen rarely I think this is a good trade.

Of course, I re-did the tests with the old fwd-h2-1.4.200-6.jar and the result were pretty unchanged, average being ~200ms for both cases. Those results from my note #6928-4 above still stand.

#10 - 12/05/2022 02:13 AM - Alexandru Lungu

version	server run	client run	indexed (ms)	unindexed(ms)
h2-1.4.200.jar	1	1	236	151
h2-1.4.200.jar	1	2	131	100

h2-1.4.200.jar	1	3	100	68
h2-1.4.200.jar	1	4	94	47
h2-1.4.200.jar	1	5	91	46
Warm avg.			**104**	**65**
h2-1.4.200.jar	2	1	296	201
h2-1.4.200.jar	2	2	113	58
h2-1.4.200.jar	2	3	97	50
h2-1.4.200.jar	2	4	100	53
h2-1.4.200.jar	2	5	91	47
Warm avg.			**100**	**52**

These are my testing results with the new jar. The times are certainly better than in #6928-7 (3 times better for unindexed is really good).

> Alexandru: I recall there were some performance testcases written for H2, can you point me where these are?

There is a thread about Performance Testing the H2 Database. The most accurate results are achieved using the testcases/uast/h2_profiling/start.p and testcases/uast/h2_profiling/legacy-tests/perf-run-all-silent.p tests (using etime). The first tests are the ones integrated in Hotel GUI.

#11 - 12/12/2022 12:19 PM - Constantin Asofiei

- Status changed from WIP to Review

- % Done changed from 0 to 100

The changes are in fwd-h2 project rev 8.

Alexandru: please review.

#12 - 12/13/2022 06:58 AM - Alexandru Lungu

I got deep in most of the changes in rev. 8 - they seem reasonable. As a side question, I see some changes from for-each structure into index based iteration: are these changes targeting performance?

#13 - 12/13/2022 07:47 AM - Constantin Asofiei

Alexandru Lungu wrote:

> As a side question, I see some changes from for-each structure into index based iteration: are these changes targeting performance?

Yes, the point is that for-each creates iterator-related objects which affect garbage collection.

#14 - 01/06/2023 03:00 AM - Eric Faulhaber

Is there anything left to do, in order for FWD to use the new H2 jar by default? Is the FWD build updated? Is the jar uploaded to the artifacts site? If not, is there any reason it should not be?

#15 - 01/20/2023 09:28 AM - Constantin Asofiei

H2's UPDATE command was profiled and improved to avoid object allocations and unnecessary iterations. This is in fwd-h2 rev 8, but fwd-h2 rev 7 shows a slowdown, so this can not be released until fwd-h2 rev 7 is fixed.

#16 - 01/20/2023 11:37 AM - Alexandru Lungu

H2 rev. 7 is mostly targeting progressively fetched (using LIMIT) multi-table AdaptiveQuery. This is not yet part of FWD, so the optimization itself doesn't have the expected impact on its own. It looks like the overhead there is much more than the actual benefit. As we have an "experimental" H2 branch, I think we can integrate the UPDATE optimizations alone in our "official" FWD-H2 branch. Otherwise, we can move rev. 7 to an even more experimental branch, until we have a resolution for #6582.

#17 - 01/20/2023 12:29 PM - Eric Faulhaber

Alexandru Lungu wrote:

> H2 rev. 7 is mostly targeting progressively fetched (using LIMIT) multi-table AdaptiveQuery. This is not yet part of FWD, so the optimization itself doesn't have the expected impact on its own. It looks like the overhead there is much more than the actual benefit. As we have an "experimental" H2 branch, I think we can integrate the UPDATE optimizations alone in our "official" FWD-H2 branch. Otherwise, we can move rev. 7 to an even more experimental branch, until we have a resolution for #6582.

Agreed. As has been suggested elsewhere, it is time to get our fork of the H2 source code base under version control. I'll open a separate task for this.

#18 - 01/20/2023 12:44 PM - Eric Faulhaber

- Related to Support #7058: get the FWD fork of the H2 code base under version control added

#19 - 01/25/2023 05:34 AM - Radu Apetrii

While working on another task, I stumbled upon an issue. With fwd-h2 rev.7 it worked fine, **but in rev.8 there seems to be a problem regarding rollback to savepoint**.

This is the test I tried in a H2 console:

```
create table tt (f1 int)
insert into tt (f1) values (1)
savepoint sv
update tt set f1 = 2
select * from tt
rollback to savepoint sv
select * from tt
```

After the rollback is executed, I was expecting to find f1 = 1, but, the value did not change, thus being equal to 2.
Note: this only happens when an update statement is executed (insert and delete work fine).

#20 - 01/25/2023 05:36 AM - Constantin Asofiei

Radu, please debug this and fix it.

#21 - 01/25/2023 10:25 AM - Radu Apetrii

- File 6928.patch added

I added a patch that fixes the problem that I found earlier (and another one that I found while testing). Long story short, in Update.update when the old row's data got updated, **the session should've logged the changes to undoLog** (in case a rollback would happen later).

The other thing that I added was to **update the database's last operation id**, so the cache would not interfere when querying the same select statement, but after the update. This was the other test that failed (a select statement, an update and then the same select -> the result was the same for both select s, even though an update happened and changed the record).

If you find any more issues, you can let me know. As far as I tested, everything seemed OK, but there might be things that I have missed.

#22 - 02/11/2023 10:59 AM - Constantin Asofiei

Rebased 6928a_h2 rev 8 from fwd-h2.trunk rev 8 - new rev 10.

#23 - 02/14/2023 06:20 AM - Alexandru Lungu

Tested 6928a_h2/rev. 10 and found a regression: the old-row stored by deltaChangeCollector inside the H2 update statement was altered by our modifications. Therefore, SELECT * FROM OLD TABLE (UPDATE TEST SET B = 3 WHERE ID = 1); was returning updated rows, instead of the old ones. Fixed this is 6928a_h2/rev. 11.

As all regressions tests now pass, I am planning to merge 6928a_h2 into trunk if no further objections.

#24 - 02/14/2023 06:23 AM - Constantin Asofiei

Alexandru Lungu wrote:

> Tested 6928a_h2/rev. 10 and found a regression: the old-row stored by deltaChangeCollector inside the H2 update statement was altered by our modifications. Therefore, SELECT * FROM OLD TABLE (UPDATE TEST SET B = 3 WHERE ID = 1); was returning updated rows, instead of the

> old ones. Fixed this is 6928a_h2/rev. 11.

Was the branch bound? I don't see rev 11 for 6928a_h2.

#25 - 02/14/2023 06:26 AM - Alexandru Lungu

Constantin Asofiei wrote:

> Was the branch bound? I don't see rev 11 for 6928a_h2.

What a quick response! My bzr was still committing when I've written #6928-23 :) Now it is finished, please try updating again.

#26 - 02/14/2023 06:52 AM - Constantin Asofiei

Thanks, the change is OK. It can be merged to trunk.

#27 - 02/14/2023 07:01 AM - Alexandru Lungu

- Status changed from Review to Test

Merged 6928a_h2 to FWD-H2 trunk including performance improvements for UPDATE statement, with in-memory and PageStore databases.
FWD-H2 trunk is now at rev. 9.

#28 - 01/19/2024 01:26 AM - Eric Faulhaber

- Status changed from Test to Closed

I am closing this. I think 11 months of testing is probably sufficient :)

Files

h2_update_perf.patch	27.8 KB	11/15/2022	Constantin Asofiei
h2-1.4.200.jar	2.24 MB	11/28/2022	Constantin Asofiei
h2_update_perf2.patch	16.4 KB	11/28/2022	Constantin Asofiei
updateperf.p	2.12 KB	11/28/2022	Constantin Asofiei
h2-1.4.200.jar	2.24 MB	12/02/2022	Constantin Asofiei
h2_update_perf3.patch	20.3 KB	12/02/2022	Constantin Asofiei
6928.patch	2.24 KB	01/25/2023	Radu Apetrii