Database - Bug #7351

Reduce SQL query string size of an INSERT INTO SELECT FROM

05/15/2023 07:06 AM - Alexandru Lungu

Status:	Closed	**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 - Bug #7330: Increase psCache size	**Closed**
Related to Database - Feature #7382: Check performance of delete from vs drop...	**Test**
Related to Database - Bug #7389: Make fast-copy statements cachable	**Closed**

History

#1 - 05/15/2023 07:10 AM - Alexandru Lungu

The COPY-TEMP-TABLE statement sometimes uses a "fast" approach if possible. It analyses table signatures (field types, indexes, etc.) and if they match, a INSERT INTO [...] SELECT FROM [...] statement is emitted. This works as a bulk select and insert.

The concern now is that such statement is very large, because it includes the field lists of the source and destination.

Please check if the wildcard syntax (tt.*) can be used instead generating the whole field list for source and destination. Test with large applications. The wildcard syntax is already used when "expanding alias" when generating query SQL.

#2 - 05/15/2023 07:10 AM - Alexandru Lungu

- Related to Bug #7330: Increase psCache size added

#3 - 05/17/2023 05:41 AM - Alexandru Lungu

- Status changed from New to WIP

- Assignee set to Dănuț Filimon

Please check FastCopyHelper and if the field list can be compressed with a simple tt.* syntax. Note that this is used exclusively for temp tables.
Also note that there are two selection lists (one for INSERT INTO and one for SELECT FROM).
Make some tests with copy-temp-table up-front and check if everything is alright (test with/without append and with/without loose-mode). Make sure fast copy is triggered (source and destination have the same signature).
Dump the SQL executed in your testcases and ensure they use tt.* syntax.

#4 - 05/19/2023 02:59 AM - Alexandru Lungu

Danut, if you already do modifications to the FastCopyHelper, I will request more things here:

- Fix the javadoc as there are many parameters which don't have the param javadoc tag. Some methods don't have the right multi-line javadoc format.
- Ensure code standard are respected. I see some inconsistencies like for (int i=0; i<from.size(); i++) with no proper spaces.

- Make sure that each "extra" table generated in the process (MASTER__PK__MAPPING, BEFORE__PK__MAPPING, LIST__INDEX__TABLE) have unique names across all fast-copy iterations. Therefore, I expect to have MASTER__PK__MAPPING_1, MASTER__PK__MAPPING_2, etc. I want to remove the recompiling of statements from H2 for good and the only tables which stand in the way are these. They are created and dropped, but share the same name. Even if cache may hit for these, the underlying prepare statement should be recompiled anyway as the physical table is brand anew.
- **EDIT**: also run testscases/uast/copy-temp-table. Do a master procedure to run all tests sequentially. The point is to run multiple copy-temp-table in the same session to check that the uid works properly.

For the second point, add a class member (lets say long uid) generated uniquely. Use a static AtomicLong sequence to ensure uniqueness of uid. Set the uid in constructor.
At this point, whenever you append the MASTER__PK__MAPPING, BEFORE__PK__MAPPING or LIST__INDEX__TABLE names, append the _{uid} identifier.

#5 - 05/19/2023 09:36 AM - Alexandru Lungu

Danut, I think I have a better idea.

My concern is that dropping tables like MASTER__PK__MAPPING_1 will make the cached statements unusable. Basically:

- INSERT INTO tt SELECT FROM tt2 JOIN MASTER__PK__MAPPING_1 is different from INSERT INTO tt SELECT FROM tt2 JOIN MASTER__PK__MAPPING_2, so these are two different prepared statements, which spoil the cache as both MASTER__PK__MAPPING_1 and MASTER__PK__MAPPING_2 are dropped right after.
- Even when we had INSERT INTO tt SELECT FROM tt2 JOIN MASTER__PK__MAPPING, this prepared statement was cached. Even if a cache hit happened, the statement was recompiled.

What if we don't drop MASTER__PK__MAPPING, but just clear it. There can't be two simultaneous fast copy-temp-table as the H2 sessions are independent from each other. Clearing the table should be enough so that the prepared statement can be re-executed without recompile. The only question I have is if clearing a table is slower than dropping it.

Please ignore #7351-4 for now and:

- compare on a dummy FWD-H2 the time difference between dropping a table and deleting all of its elements. Try on a big table (~100 columns) with several indexed (~5) and several rows (~1000)
- If results are OK, replace DROP tt with DELETE FROM tt. Make the creation with CREATE IF NOT EXISTS. Intensively test, even with multi-users.
- If everything is OK, I will add a neverRecompile flag and do some performance tests.

#6 - 05/22/2023 03:00 AM - Dănuț Filimon

Alexandru Lungu wrote:

> - compare on a dummy FWD-H2 the time difference between dropping a table and deleting all of its elements. Try on a big table (~100 columns) with several indexed (~5) and several rows (~1000)

I tested 1k, 10k and 100k records in H2 and the performance of DROP is better than DELETE.

Records	DROP (ms)	DELETE (ms)
1000	2.8	66.8
10000	89	112
100000	83.2	923.6

Should #7351-5 still be considered in this case? And should I still work on unique names the extra generated tables if not?

Regarding the INSERT INTO [...] SELECT FROM [...] statement. Something like

```
insert into tt4 ( recid,_multiplex,_errorFlag,_originRowid,_datasourceRowid,_errorString,_peerRowid,_rowState,
f1) direct sorted select next value for seq_3,?,_errorFlag,_originRowid,_datasourceRowid,_errorString,_peerRow
id,_rowState,f1 from tt3 use index (idx_mpid__tt3__3) where _multiplex = ? order by _multiplex,recid {1: 2, 2:
 1};
```

can be written as

```
insert into tt4 direct sorted select next value for seq_3,?,tt3.* EXCEPT (tt3.recid, tt3._multiplex) from tt3
use index (idx_mpid__tt3__3) where _multiplex = ? order by _multiplex,recid {1: 2, 2: 1};
```

At the moment, I only modified FastCopyHelper.copyTable to generate such statement. I still need to test append and loose-mode and make the necessary modifications.

#7 - 05/22/2023 03:54 AM - Alexandru Lungu

> At the moment, I only modified FastCopyHelper.copyTable to generate such statement. I still need to test append and loose-mode and make the necessary modifications.

The INSERT INTO SELECT from changes are good to go in any form. Having shorter SQL is better all the way.

> Should #7351-5 still be considered in this case? And should I still work on unique names the extra generated tables if not?

This is a really big gap between DROP and DELETE. I wonder if we can take this into account for our delete use-cases (which use multiplex). Anyway, continue with single uid. Even if I don't like having unusable cached prepared statements, this is something that already happens, so uid won't bring any harm.

However, we need to think of a better mean of caching these copy-temp-table operations. AFAIK, these intermediate tables are used only with normalized extent, APPEND or BEFORE modes - so we have still have the common-case operations properly cached.

Danut, I created 7351a and 7351a_h2. Please create a neverRecompile connection option (much like always recompile) in 7351a_h2 and test it. We will leave your implementation of dependency checks (in needsRecompile for persistent H2 cases. This flag should be used only for in-memory tables for now! Feel free to update H2Helper.setCommonInMemoryProperties to include neverRecompile.

Mark #7351 your top priority (both SQL compression and uid / neverRecompile features).

#8 - 05/23/2023 06:11 AM - Dănuț Filimon

Committed 7351a/rev.14576. Reduced sql size of the INSERT INTO [...] SELECT FROM [...] statement by using wildcard when possible.

The changes include:

- Added AtomicLong uid value used to generate unique names for MASTER__PK__MAPPING, BEFORE__PK__MAPPING, LIST__INDEX__TABLE, the unique name is generated using assembleUidMapping method.
- Added boolean ALLOW_WILDCARD used to toggle between the old and new creation of insert based on select statements. Note that even if this value is set to true, it will not be used when the copy is in **loose mode** because of the possible mismatch in number of parameters or order of the columns.
- Created getSrcTempWildcard and getSrcMappedWildcard that builds the source table wildcard string. The columns recid and _multiplex are excepted from the wildcard selection.
- The insert statement can't use wildcard when in loose mode so executeCopy(), copyBeforeTable(), copyMasterTable(), copyTable() and safeCopyTable() receive an additional looseCopy parameter and it is used in copyTable and safeCopyTable methods.
- Added missing javadoc (@param, @throws PersistenceException), fixed formatting.

Please review.

#9 - 05/24/2023 04:21 AM - Alexandru Lungu

- % Done changed from 0 to 70

Review of 7351a/rev.14576

- I don't think we need to lazily initialize uid. It is ok to have it only once as it is static. This way, we don't have to check each type we do a fast copy if the uid was created or not.
- ALLOW_WILDCARD should be replaced with P2JH2Dialect.allowSelectWildcard, which is already implemented. Also you must ensure there are no computed columns using DmoMeta.hasComputedColumns. Please take example from FqlToSqlConverter.expandAlias. If we want to disable wildcards, we can do this globally using allowSelectWildcard.
- Please align arguments of getSrcMappedWildcard and appendColumnName in safeTableCopy and copyTable.
- You deleted .append(ReservedProperty._ROWSTATE.column) from getSrcMappedColumns. Should it be that way?
- **You are not using uid right!** If multiple users call incrementAndGet, when you will compose the name of the table with uid.get(), you will retrieve the latest id. That is, if 10 users do the fast-copy in the same time, the use of this uid is messed up (a single user can retrieve different values at different times). The goal is to store the id of incrementAndGet and use it exclusively with that fast copy helper. No .get calls at different point of times for a single user. Please fix this - it can cause "table not found" regressions very easily in multi-user environments.

Danut, please integrate the neverRecompile flag with 7351a_h2 and add the required connection option to FWD in 7351a. I will test the FWD and FWD-H2 changes simultaneously. Do in-depth tests with a large customer application after you apply neverRecompile.

Mark this task your top priority! I would like to have this completely reviewed, tested and profiled by the end of this week. Thank you!

#10 - 05/24/2023 07:25 AM - Dănuț Filimon

Committed 7351a_h2/rev.18. Added NEVER_RECOMPILE property.
Committed 7351a/rev.14577. Fixed uid usage, replaced ALLOW_WILDCARD, added back .append(ReservedProperty._ROWSTATE.column) since I deleted it by mistake and added NEVER_RECOMPILE to H2 connection for single-user databases.

I can't test any large customer application due to logging issues. Once the latest version is released, I will be able to do so.

#11 - 05/24/2023 08:45 AM - Alexandru Lungu

Danut, can you port this changes to 7026d and do the tests with it? 7026d isn't rebased yet to use the latest logging changes, so you may have a good run with it on a large customer application. I think is for the best to keep this change to 7026d anyway.
Thank you.

#12 - 05/24/2023 08:47 AM - Alexandru Lungu

In fact, please commit everything there except the NEVER_RECOMPILE flag in FWD for temp tables. You will commit that after we build and release a new FWD-H2 version. We will bump the version to 7026d and afterwards you can add the connection option. Otherwise, 7026d will break.

#13 - 05/24/2023 09:20 AM - Dănuț Filimon

Committed 7026d/rev.14576. Added the changes from 7351a rev. 14576 & 14577 without the NEVER_RECOMPILE property.

#14 - 05/24/2023 09:57 AM - Dănuț Filimon

I had no problems testing a customer application for ~20 minutes with 7026d/rev.14576 and NEVER_RECOMPILE property.

#15 - 05/24/2023 10:07 AM - Alexandru Lungu

Dănuț Filimon wrote:

> I had no problems testing a customer application for ~20 minutes with 7026d/rev.14576 and NEVER_RECOMPILE property.

This is great. Having a stable 7026d right now is important.

#16 - 05/25/2023 04:17 AM - Alexandru Lungu

- % Done changed from 70 to 80

Please do a memory check on psCache. That is, please check if the memory is less now than before on a large customer application.
You can use VisualVM > Monitor > Heap Dump and sort Retained sizes to compute the retained memory. Search for TempTableDataSourceProvider.Context. Each such instance stores a psCache. You can even check what is inside the cache. Please compute: the size of the largest UnclosablePreparedStatement, the total size, the avg. size of an UnclosablePreparedStatement and analyze the largest 10 UnclosablePreparedStatement. State what kind of statement they are (select / insert-select / update / select with join, etc.)

You can save on the disk the memory snapshots to analyze later.

#17 - 05/25/2023 05:21 AM - Alexandru Lungu

Review of 7351a_h2/rev.18

I am OK with the changes.
Rebased and merged to FWD-H2 trunk as rev. 20. 7351a_h2 was archived.

#18 - 05/25/2023 09:11 AM - Dănuț Filimon

Committed 7026d/rev.14580. Added a flag that enables uid, it's usage is now disabled by default.

After doing a memory test on psCache and analyzing the heap dump obtained with VisualVM with and without modifications from the current issue, I remarked that there are a lot more statements stored that use intermediate table with uid, which increases the memory used by psCache (~2-3x). That's why I added a flag to enable the uid in the future. The NEVER_RECOMPILE flag will also remain, but will also not be added to the H2 connection url yet.

The changes without the uid also use more memory, but not by much. I will continue to take a look at UnclosablePreparedStatement and see what statements use more memory.

	Retained (B)	Retained 7026d/rev.14580 (B)	Difference (%)
TempTableDataSourceProvider$Context	5094568	6358320	+24.805%

#19 - 05/25/2023 09:13 AM - Alexandru Lungu

Danut, note that 7026d was rebased with latest trunk. Please update.

#20 - 05/26/2023 03:54 AM - Dănuț Filimon

I took a look at UnclosablePreparedStatement and gathered the following data:

	Count	Total Retained Size (B)	Average Size (B)
UnclosablePreparedStatement (Old)	2529	8293704	3279.44
UnclosablePreparedStatement (7351)	2529	8068360	3190.336

For the 10 largest statements I have found the following while testing without any changes:

Type	Retained Size (B)
select	35072
select	32392
select	29592
select	28808
insert select	27432
select	26984
select	26600
select	26528
select	21680
insert select	20472

There are two INSERT INTO [...] SELECT FROM [...] and eight select statements.

After applying the changes from #7351, the insert statements are no longer present in the top 10:

Type	Retained Size (B)
select	35072
select	32312
select	29592
select	28824
select	26984
select	26664
select	26528
select	21680
select	15904
select	14720

I managed to find the exact matching INSERT INTO [...] SELECT FROM [...] statements in the two heap dump snapshots, here we can see that there is an improvement in the size used by the statement:

Nr.	Statement	Retained Size - Old (B)	Retained Size - New (B)	Difference
1	insert select	27432	13536	-50.656%
2	insert select	20472	10784	-47.323%

Note that even if I mention #7351, the tests were done with 7026d.

#21 - 05/26/2023 04:11 AM - Alexandru Lungu

- Status changed from WIP to Review

- % Done changed from 80 to 100

7026d was rebased and merged into 7156a as patch. ETF tests passed; everything looks right.

For the uid of fast copy, we should think more about it. Anyway, we can close this and focus on #7382.

#22 - 05/29/2023 05:19 AM - Alexandru Lungu

- Related to Feature #7382: Check performance of delete from vs drop table in H2 added

#23 - 05/29/2023 05:19 AM - Alexandru Lungu

- Related to Bug #7389: Make fast-copy statements cachable added

#24 - 05/29/2023 05:20 AM - Alexandru Lungu

- Status changed from Review to Test

Merged 7026d to trunk as rev. 14587.
Side-issues can be discussed in #7382 and #7389.
This can be closed.

#25 - 05/29/2023 12:57 PM - Eric Faulhaber

- Status changed from Test to Closed