Database - Bug #6812

fix FastFindCache to use a shared instance per persistent database

10/05/2022 02:56 PM - Constantin Asofiei

Status:	Internal Test	**Start date:**	
Priority:	High	**Due date:**	
Assignee:		**% Done:**	90%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD		

Description

History

#2 - 10/05/2022 02:59 PM - Constantin Asofiei

We use RecordIdentifier in FastFindCache having the table identifier a simple string with the record's implementation class. This is not correct - you can have the same table in multiple databases (think meta table, like _file), and the result will be incorrect:

- run a FIND on _file for a database where no record exists; this gets cached
- run a FIND on _file for a database where a record does exist - FWD will incorrectly use the cached value for the previous FIND, as the cache has no knowledge about the record's database.

This needs to be fixed in 6129a.

#3 - 10/05/2022 03:07 PM - Eric Faulhaber

Does it make more sense to qualify the table name with database name, or to have a separate cache for each database?

#4 - 10/05/2022 03:56 PM - Constantin Asofiei

Eric Faulhaber wrote:

> ... or to have a separate cache for each database?

Good point. It needs to be per database and get invalidated if a database disconnects.

#5 - 10/05/2022 04:09 PM - Constantin Asofiei

Eric, the FastFindCache looks like is only for temp-tables, and not other databases (like meta). So my assumption is incorrect...

The customer reported a test like this:

```
def var tname as char.
def var fname as char.
```

```
tname = "book".
fname = "isbn".

find fwd._file where fwd._file._file-name = tname no-lock no-error.
if available (fwd._file)
then do:
  find fwd._field of fwd._file where fwd._field._field-name = fname no-lock no-error.
  if available fwd._field
    then message "1 found" fwd._field._label.
    else message "1 not found".
end.
else message "1 not found".

find fwd._field where fwd._field._field-name = fname no-lock no-error.

if available fwd._field
then message "2 found" fwd._field._label.
else message "2 not found".
```

where in the first case the find fwd._field of fwd._file where fwd._field._field-name = fname no-lock no-error. sometimes returns a record and sometimes not. And sometimes even _file meta table does not find a record.

My assumption was that this being intermitent, the FastFindCache may be at fault... but this is only for temp-tables.

Do you have any idea of any other kind of cache which would interfere with this? In customer's case, the database is referenced via an alias, and the program is called for different databases.

#6 - 10/05/2022 04:11 PM - Constantin Asofiei

Constantin Asofiei wrote:

> Eric, the FastFindCache looks like is only for temp-tables, and not other databases (like meta). So my assumption is incorrect...

I take this back, I read the code wrong - FastFindCache is not being used for meta databases, in RAQ.initialize:

```
      if (!dmoMeta.isMeta() && index != 0)
      {
         this.ffCache = FastFindCache.getInstance(buffer.isTemporary());
      }
```

#7 - 10/05/2022 06:36 PM - Greg Shah

> FastFindCache looks like is only for temp-tables

It isn't used for permanent databases?

#8 - 10/05/2022 07:22 PM - Ovidiu Maxiniuc

The factory makes sure the permanent databases use a commonly shared instance of FastFindCache while for the temp-tables a context-local is used.

There should be no problems with temp-tables instances. Maybe we should do the same for the _meta?

#9 - 10/06/2022 01:32 AM - Constantin Asofiei

FastFindCache is now created for both temp-table and permanent databases. The problem is that for permanent (or meta, if we enable it) databases, the cache does not differentiate between tables with the same structure from different databases.

Regarding the original test in note #6812-5 - I don't see how FastFindCache can be involved, as this is not enabled for meta tables. So, does anyone have any idea what other cache would be used in this test?

#10 - 10/08/2022 06:26 PM - Eric Faulhaber

Ovidiu Maxiniuc wrote:

> The factory makes sure the permanent databases use a commonly shared instance of FastFindCache while for the temp-tables a context-local is used.
>
> There should be no problems with temp-tables instances. Maybe we should do the same for the _meta?

Igor explicitly disabled FFC for _meta due to some problematic behavior. I wonder now whether that behavior essentially was this defect. Igor, do you recall what the issue was that led you to disable the cache for metadata tables?

#11 - 10/28/2022 03:42 PM - Eric Faulhaber

- Subject changed from fix FastFindCache to use the database.table as key instead of just table to fix FastFindCache to use a shared instance per persistent database

Changed the issue description to match the point made in #6812-3 and #6812-4.

#12 - 02/16/2023 03:26 PM - Eric Faulhaber

- Assignee set to Boris Schegolev

Boris, please look at the FastFindCache implementation and ask any questions you may have as it pertains to this task.

#13 - 02/23/2023 04:40 PM - Boris Schegolev

- File 6812-fast-find-cache.diff added

I am attaching a suggested solution. It's somewhat raw (I had to make some elements public), but I can fix that for the final version. For now I just need to know if that approach makes sense at all and if it potentially solves the problem described. Please, let me know. Thank you!

#14 - 02/23/2023 05:17 PM - Eric Faulhaber

Please create a 6812a branch from trunk and apply the diff. This will make it easier to review in context.

I've only done a cursory review so far, but please use the Database object instead of the database name. The database name will be the same in multi-tenant (database-per-tenant) situations.

#15 - 02/24/2023 09:18 AM - Boris Schegolev

I pushed the patch as revision 6812a/14485, working on better implementation (Database object as key, cleanup, etc.)

#16 - 02/24/2023 03:12 PM - Boris Schegolev

- Status changed from New to WIP

I converted the code to use Database object directly and reverted changes I made initially to visibility modifiers. See revision 6812a/14486.

I just need to figure out one call in SavepointManager. I didn't find any suitable object here that would give me the Database.

#17 - 03/06/2023 05:10 PM - Boris Schegolev

- Status changed from WIP to Review

- % Done changed from 0 to 90

I prepared a change for the last missing bit (SavepointManager) - see revision 6812a/14487. It needs more testing, but the code can reviewed already.

#18 - 04/19/2023 04:00 PM - Eric Faulhaber

Code review 6812a/14487:

Sorry for the delay in reviewing. A lot of changes have gone in to trunk in the meantime; please rebase. Hopefully, this doesn't impact your changes too much.

Please:

- add missing files header entries;
- add missing javadoc for any new constructs;
- update the FastFindCache class javadoc to describe the organization of cache instances by permanent and temporary databases.

I don't think we need to store a reference to a Database instance in each SavepointManager$Block instance. Instead, why not just pass the Session's Database instance into SavepointManager$Block.invalidateFastFind from SavepointManager.rollback? This is the only place the Database is needed.

#19 - 05/04/2023 03:53 PM - Boris Schegolev

- Status changed from Review to WIP

#20 - 05/11/2023 07:05 PM - Boris Schegolev

- Status changed from WIP to Review

I pushed the requested changes as revision 6812b/14567. This is a new branch based on current trunk, I'll drop 6812a. Please, review.

#21 - 05/12/2023 08:09 AM - Greg Shah

I'll let Eric review this. In the future, please do not create a new branch instead of rebasing. That typically loses all continuity with the work from the branch (it is "flattened" into a single revision and it doesn't map to any discussion in the task). Rebsing is quite easy to do. Use it.

#22 - 05/12/2023 01:13 PM - Boris Schegolev

Greg Shah wrote:

> I'll let Eric review this. In the future, please do not create a new branch instead of rebasing. That typically loses all continuity with the work from the branch (it is "flattened" into a single revision and it doesn't map to any discussion in the task). Rebsing is quite easy to do. Use it.

I understand the consequences and that it's easy to do (usually). It didn't work out in my case and it was just an implementation commit + review fixes, I thought it would be more efficient this way.

#23 - 05/12/2023 01:49 PM - Eric Faulhaber

Code review 6812a/14567:

Looks good. I did a little format cleanup and committed to rev 14568. I did not update the file headers for these minor edits.

Since trunk has moved ahead today, another rebase is needed.

What testing has been done? The changes are pretty straightforward, but this code is hit very often and I don't like committing anything to trunk without some regression testing.

#24 - 05/12/2023 05:10 PM - Boris Schegolev

Eric Faulhaber wrote:

> Since trunk has moved ahead today, another rebase is needed.

Rebased, conflicts resolved, headers updated. Revision is 6812b/14573.

> What testing has been done? The changes are pretty straightforward, but this code is hit very often and I don't like committing anything to trunk without some regression testing.

Just testing and debugging on a running application. I haven't done any load testing to validate performance and concurrency. On the other hand, the code uses ConcurrentHashMap with atomic operations, so there should be no issues here.

#25 - 09/15/2023 10:28 AM - Greg Shah

- Assignee deleted (Boris Schegolev)

How do we move this along?

#26 - 11/29/2023 10:59 AM - Eric Faulhaber

- Status changed from Review to Internal Test

Constantin, does it make sense to regression test this with the large POC?

Files

6812-fast-find-cache.diff	6.66 KB	02/23/2023	Boris Schegolev