Database - Feature #6496

add equivalent support for -trig command line option

06/06/2022 12:25 PM - Greg Shah

Status:	Test	**Start date:**	
Priority:	High	**Due date:**	
Assignee:	Alexandru Donica	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**vendor_id:**	GCD

Description

History

#2 - 06/06/2022 12:27 PM - Greg Shah

-trig <tPath> defintes the location (directory/library) to find triggers.

Partially supported already if the path is added to PROPATH.

Notes:

1. this might improve performance if triggers are looked up in a single location instead of scanning the whole PROPATH.
2. libraries are some .pl files (native code?, .r code?, else?)

Item 2 won't matter for FWD, but if this can improve performance we should implement it.

#3 - 06/26/2023 04:34 AM - Alexandru Lungu

- Assignee set to Andrei Bălteanu

#4 - 09/12/2023 05:01 AM - Alexandru Lungu

- Assignee changed from Andrei Bălteanu to Alexandru Donica

- Status changed from New to WIP

Alex, please handle this now. There is some insight in spr and proghand documentations, but quite minimal.
Attempt to prioritize performance when doing the look-up for triggers using -trig.
You can do some initial investigation around DatabaseTriggerManager.getSchemaTrigger.

Note that legacy database connection options we support are usually stated in directory.xml. I guess this -trig option can be added as a database container sub-node. Note that this option should be "per-database".

#5 - 09/21/2023 03:19 AM - Alexandru Donica

- % Done changed from 0 to 100

- Status changed from WIP to Review

Commited branch #6496a on rev. 14743 and 14744,
with necessary changes to support -trig parameter for searching a specific folder and not the entire PROPATH.

I tested the code and compared the time to find a 'trigger class' with a PROPATH where the folder with the triggers was not among the first segments, with and without the 'trig' parameter. This difference grows considerably when testing without 'trig' and the desired folder is placed further down the PROPATH segments.

With the 'trig' parameter, it's as if the desired directory is placed first in the PROPATH list, and without the 'trig' parameter, the time is very similar to the search time from before the changes were implemented, where it would automatically search through the PROPATH.

In directory.xml, the 'trig' node should be found at 'database/{db_name}/p2j/trig', as shown below for a database called 'fwd'.

This search is only done once per database and cached when it is first needed.

I modified existing code to support searching not only through the PROPATH, but through an array of paths, which, when not provided, becomes the PROPATH.

```
        <node class="container" name="database">
          <node class="container" name="fwd">
            <node class="container" name="p2j">
              <node class="string" name="schema">
                <node-attribute name="value" value="fwd"/>
              </node>
              <node class="string" name="trig">
                <node-attribute name="value" value="db-triggers"/>
              </node>
              ...
```

#6 - 11/02/2023 01:23 AM - Eric Faulhaber

Code review 6496a/14743-14744:

The configuration is causing me some confusion, in that it can be specified for each database; however, multiple databases can use the same schema. The schema name is the key being used for the dbTrigPathMap in DatabaseTriggerManager:

```
         String databaseName = DmoMetadataManager.getDmoInfo(dmoIface).getSchema();

         trigPath = dbTrigPathMap.computeIfAbsent(databaseName, (k) ->
         {
            ...
```

Here, the variable databaseName is a misnomer; what is being stored is actually the schema name. So, we can potentially have different -trig configurations for multiple databases, and the first one read will be stored in the map. The others will be lost. If we want to scope these to physical database instances, we should use a Database object as the key, rather than the schema name associated with the DMO.

This brings up a further point w.r.t. the scope of the -trig parameter: should we be mapping it to the database instance? In OE, -trig is a client connection parameter, which can be different for every client connection. It seems we map propath to a particular application server instance. Do we want to do the same here, or do we want to force one set of -trig paths per database instance? The current implementation does not match either approach.

Aside from the above, the changes to DatabaseTriggerManager seem logically correct, given the configuration approach taken. There are a few minor formatting / coding standard / typo issues:

- Remove the hard tab at line 72 and fix the typo: "seach".
- Use wildcards for import statements, unless resolving an import conflict.
- Follow coding standards w.r.t. the use of spaces (e.g., use if (... instead of if(...).

There are many changes to SourceNameMapper. Alexandru (Lungu) or Danut: can you please confirm these are consistent with / not conflicting with any changes in approach being taken for #6649?

#7 - 11/02/2023 05:54 AM - Dănuț Filimon

Eric Faulhaber wrote:

> There are many changes to SourceNameMapper. Alexandru (Lungu) or Danut: can you please confirm these are consistent with / not conflicting with any changes in approach being taken for #6649?

There is an additional parameter for convertName in #6649 because of fastConvertName which needs to work with WorkArea, it will pass the WorkArea to the old convertName implementation when it fails (to avoid additional costly calls), but it should not pose a big problem. There are no other conflicts from what I can see.

#8 - 11/02/2023 10:14 AM - Eric Faulhaber

OK, so then other than fixing the typo/coding standard issues, the only thing left to decide is the scope we want to support for the -trig parameter.

If we want to allow full compatibility and differentiation **by client connection**, this will require storing trigger path information at a context-local level, and some refactoring of the way the configuration is defined in the directory. I am not sure exactly what the directory entries would need to look like.

If we want to allow differentiation **by physical database**, the configuration can stay as it is today, but we will need more information to be passed through to DatabaseTriggerManager.getSchemaTrigger. Currently, the following parameters are passed to this method:

```
DatabaseEventType det
Class<? extends Buffer> bufferCls
String property
```

This is enough to determine the schema being used, but not the physical database. This would also require some refactoring of the methods which call this method (and the methods that call those methods, and so on...), because not all of them have the required information to determine physical database either. It seems like this would be a good bit of API refactoring work.

If we want to allow differentiation **by schema**, the current logic will mostly work, but there must be a change to the path of the directory lookup, and

the organization of the configuration in the directory (would have to be stored in a schema node, which does not currently exist, AFAIR, rather than in the existing database node). Currently, there is an assumption that schema name and database name are the same, but this is not a valid assumption; there is a many-to-one relationship of database name to schema name.

Greg, Constantin, Ovidiu: which scope makes sense to you? I suppose there is also the option of matching the way we configure the propath.

#9 - 11/02/2023 10:34 AM - Greg Shah

We definitely cannot set this **per database**. In most cases, customers have a single propath for all clients and would want all the processing the same (the -trig directory the same) **per schema**.

The problem is that there are some customers who definitely will have **per client** (or likely **per appserver**) settings. We need to make this a client-session parameter configurable in the runtime parameters section and passable as a command line parameter. Theodoros can advise on the implementation.

#10 - 11/02/2023 10:39 AM - Greg Shah

BTW, it doesn't seem like this setting in OE has any relationship to a given schema or database instance. It is just a session-level fast path for all trigger lookups in preference to the propath.

#11 - 11/02/2023 10:49 AM - Eric Faulhaber

In speaking with Greg about this, I also realized that I was not considering session triggers during my review. The OE docs mention no distinction between session and schema triggers for this feature.

Ovidiu, the session trigger mechanism seems quite different from the schema trigger lookup. Is the propath currently involved in resolving a session trigger (i.e., is a session trigger purely a block-oriented concept, or is there a notion of a session trigger procedure, which could have a configurable/overridable location)? In other words, does the -trig client connection parameter affect this resolution?

#12 - 11/02/2023 01:21 PM - Theodoros Theodorou

Greg Shah wrote:

> The problem is that there are some customers who definitely will have **per client** (or likely **per appserver**) settings. We need to make this a client-session parameter configurable in the runtime parameters section and passable as a command line parameter. Theodoros can advise on the implementation.

You can add a startup parameter in StartupParameters.java. Each client has its own instance of this object. You can set the property in client.xml as:

```
<client>
   <cmd-line-option trigger-location="..."/>
</client>
```

In StartupParameters.java, you will need to add something like:

```
private String triggerLocation;
...
triggerLocation = bc.getString(client, gname, "trigger-location", null);
...
triggerLocation = getParamFromDirectory(triggerLocation, "trigger-location");
...
getParameterString(parameters, triggerLocation, "-trig");
```

Greg, please confirm that the names are ok. Also, I would like to suggest merging 7732a to avoid merge conflicts.

#13 - 11/02/2023 04:59 PM - Ovidiu Maxiniuc

Eric Faulhaber wrote:

> In speaking with Greg about this, I also realized that I was not considering session triggers during my review. The OE docs mention no distinction between session and schema triggers for this feature.
> Ovidiu, the session trigger mechanism seems quite different from the schema trigger lookup. Is the propath currently involved in resolving a session trigger (i.e., is a session trigger purely a block-oriented concept, or is there a notion of a session trigger procedure, which could have a configurable/overridable location)? In other words, does the -trig client connection parameter affect this resolution?

No, session triggers are not looked up in any path. They are simply unnamed blocks defined within the code of another procedure/function/method. -trig will not affect them so that why there is no mentioning in manual.

My vote goes with Greg. Since the -trig is a client side parameter, a customer expects the triggers for all his databases to be found at this location. A scenario I can think of is an environment with multiple users, each with his own -trig folder. OTOH, the schema triggers are, by default, CRC-ed and the signature is saved to (_meta ?) database. If it exists and the CRC fails, a fatal condition is raised and the execution halts. Yet, we do not support this feature.

#14 - 11/13/2023 05:42 AM - Alexandru Lungu

- Priority changed from Normal to High

I am moving this to High priority as it may grant us some performance improvement that we really need now. Alexandru [ad], please focus on this matter when available.

#15 - 11/13/2023 05:43 AM - Alexandru Lungu

- Status changed from Review to WIP

- % Done changed from 100 to 60

#16 - 11/17/2023 07:48 AM - Alexandru Lungu

Any progress here? Mind that we need a major performance drop for the upcoming week, and the addition of -trig may provide part of the boost we are looking for.

#17 - 11/17/2023 07:50 AM - Alexandru Donica

I have been working on this little by little between other high priority tasks. I intent to put more focus on this task. Will try to update soon.

#18 - 11/21/2023 03:59 AM - Alexandru Donica

- % Done changed from 60 to 100

- Status changed from WIP to Review

I moved the trigger location parameter to StartupParameters as adviced. I also removed the map that cached the trigger locations, since now they are retrieved from SessionUtils._startupParameters(), and are 'per client'. Other than the method of "getting" the trigger location, the code is the same.
If i missed anything, let me know and I will update as soon as I see the review.
Commited on branch 6496a rev 14745.

#19 - 11/21/2023 08:09 AM - Greg Shah

Theodoros: Please review.

All: We are trying to merge 7732a before 6496a since it has changes related to the startup parameters. Or perhaps we should merge 7732a into 6496a. Before we merge, we must resolve all the project cfg changes (see #8063).

#20 - 11/21/2023 05:20 PM - Theodoros Theodorou

Greg Shah wrote:

> Theodoros: Please review.

The changes look good.

> All: We are trying to merge 7732a before 6496a since it has changes related to the startup parameters. Or perhaps we should merge 7732a into 6496a. Before we merge, we must resolve all the project cfg changes (see #8063).

Please proceed with merging 6496a to trunk. I will handle the rest. There is no need to complicate things and involve more people.

#21 - 11/21/2023 05:23 PM - Greg Shah

Is there more testing needed for this branch?

Theodoros: Is anything in 6496a dependent on the changes in 7732a?

#22 - 11/21/2023 05:33 PM - Theodoros Theodorou

Greg Shah wrote:

> Is there more testing needed for this branch?

I haven't tested this branch. Please let me know if you need me to test it.

> Theodoros: Is anything in 6496a dependent on the changes in 7732a?

No, there are no dependencies in 7732a. It can be merged. I will need to make some changes in 7732a after this branch is merged to trunk.

#23 - 11/22/2023 04:08 AM - Alexandru Lungu

I am giving it a shot with some performance tests now - if everything is OK, I will merge it.

#24 - 11/22/2023 07:42 AM - Alexandru Lungu

Alexandru [ad], just a notice. Is is possible to state a -trig relative to the propath? Thus, if you have a trigger in a/b/trig.t and you have:

- PROPATH set on a/
- -trig set on b/
- the trigger is defined simply as trig.t

Will it work? Thus, will the PROPATH combine with the trig?

#25 - 11/22/2023 09:55 AM - Alexandru Lungu

- Another question: the prefixes (String[] prefixes = new String[] {"./", ".\\", "../", "..\\"};) should be checked only if we have -trig. If we don't have such option, then the following seems wrong:

  ```
  javaClassName = SourceNameMapper.convertNameToClass(procedure, new String[]{"."});
  ```

  Why constraint the trigger searching only to "."? Shouldn't it honor the propath?
- Maybe short-circuit the prefix check if the first char is not ..
- Lastly, are we allowed to have multiple paths to -trig (like a/:b/)?
- SourceNameMapper.convertName has >110 characters; double check the spaces

My point overall is that this feature will be merged, but once we merge the SourceNameMapper cache, this optimization won't be visible at all. Even if we are now iterating less paths (only trig instead of the full propath), the cache will cut out that part entirely. Overall, this implementation is to offer full support for -trig conceptually, but performance-wise, it is (or will be) a no-op.

#26 - 11/22/2023 10:00 AM - Alexandru Donica

I did some more tests in 4GL with the propath now that I may not have done a while ago, or I did not complete them properly for some specific reasons.
However I see now that in 4GL, it searches for the trigger location in every folder in propath, and uses the first one it finds, and shows error if it does not find any.
Initially i was under the impression that it searches for the trigger location only in the ROOT directory. But it was only a coincidence as the ROOT dir was part of the propath.
I will make the modification in fwd now.

#27 - 11/22/2023 10:18 AM - Alexandru Lungu

Alexandru Donica wrote:

> I did some more tests in 4GL with the propath now that I may not have done a while ago, or I did not complete them properly for some specific reasons.
> However I see now that in 4GL, it searches for the trigger location in every folder in propath, and uses the first one it finds, and shows error if it does not find any.
> Initially i was under the impression that it searches for the trigger location only in the ROOT directory. But it was only a coincidence as the ROOT dir was part of the propath.
> I will make the modification in fwd now.

Please double-check this. It means that adding support to this won't actually provide any performance gain. On the contrary, it will add overhead. This is fine as long as it matches 4GL's behavior, but this is not the time to introduce slow-downs, so make sure it is a feature we really need to get it implemented.

#28 - 11/22/2023 10:59 AM - Greg Shah

> However I see now that in 4GL, it searches for the trigger location in every folder in propath, and uses the first one it finds, and shows error if it does not find any.

Are you saying that the -trig path is not a "fastpath" but is instead a kind of text to append to each one of the propath segments?

#29 - 11/22/2023 11:06 AM - Greg Shah

We should consider an approach similar to #1970, where we changed ControlFlowOps and the converted code to cache the program that is being executed. We do the lookup once and cache the result. Certain events like assigning the propath cause the cache to be invalidated. Otherwise, we get to avoid the lookup completely. This is the common case, since propath assignments are rare.

In this case, I think the cache could be held in the runtime instead of the converted code. That would make it easier to implement. It should provide a performance boost to do it this way and the result should be the same as the dynamic approach.

#30 - 11/23/2023 03:05 AM - Alexandru Lungu

Greg, what you are saying is a side-effect of #6649. Every convertName (or fastConvertName) is added to one of the global shared caches; all global caches are inside a map with the propath as key. We can rather extend the solution in #6649 so that the cache also considers trig.:

- Instead of storing the PROPATH as key, we can store a pair (PROPATH + trig). If we are searching for a trigger, search for (current-propath, trig-location). If we are searching for a program, use (current-propath, null).
- This is easy to extend in #6649.
- The cache is shared cross-session, so this is a bonus.

  Are you saying that the -trig path is not a "fastpath" but is instead a kind of text to append to each one of the propath segments?

Alexandru [ad], please confirm this with a test-case and explanation. We need to make sure this is the right case. I have a feeling that this holds. Greg, in #6447, note that the customer sets the -trig to a relative path that is not from the root of the project. However, the folder can be found after doing the PROPATH resolution.

#31 - 11/23/2023 04:49 AM - Alexandru Donica

So in 4GL it definitely tries to append the trig location to every propath folder, because when I reordered the propath paths, a different trigger would be called (the first one found). However, it searches the root directory even after I seem to have removed it from propath (I even printed the propath and did not find the current root project directory there). However, it searches the PROPATH first and then if it does not find the trig location using the propath searches the root dir.

#32 - 11/23/2023 04:57 AM - Alexandru Lungu

- Status changed from Review to WIP

- Assignee changed from Alexandru Donica to Dănuț Filimon

Alexandru Donica wrote:

> So in 4GL it definitely tries to append the trig location to every propath folder, because when I reordered the propath paths, a different trigger would be called (the first one found). However, it searches the root directory even after I seem to have removed it from propath (I even printed the propath and did not find the current root project directory there). However, it searches the PROPATH first and then if it does not find the trig location using the propath searches the root dir.

In this case, I will reassign this to Danut and let him append this functionality in #6649. Alexandru [ad], please provide assistance to Danut on the edge cases.

#33 - 11/23/2023 07:54 AM - Alexandru Donica

- Assignee changed from Dănuț Filimon to Alexandru Donica

Small observation.
I had tried in progress developer studio to remove all from propath (right click on project, project settings, propath, remove path, remove path etc) until there are no paths there. But when I print it in code, it still shows many default paths in PROPATH (including root dir that I mentioned in my comment above).
So it DOES concatenate every propath string to trigger location and searches there, and if it doesn't find it shows error. Also, it searches ONLY using trig path. (I mean that in case it does not find with trig path, it doesn't try searching in PROPATH but without trig path concatenated). I mention this just to have a more proper record of the behavior of -trig written here.
One last test case regarding the interaction -trig path has with the propath (if it does in this case) I don't know how to test, is if the -trig path is absolute. (because I can't create a folder "C:" for example in the root dir of the project, which is in the propath).

#34 - 11/23/2023 08:52 AM - Greg Shah

> However, it searches the root directory even after I seem to have removed it from propath (I even printed the propath and did not find the current root project directory there).

I assume you mean current directory, not the file system root directory. Overall, you are correct that the 4GL does indeed have some default propath entries can cannot be removed.

> what you are saying is a side-effect of #6649

If we implement the solution in #6649, we still have to do a minimal cache lookup for each trigger execution. We can avoid that if the trigger manager would cache the trigger program along with the trigger definition itself.

Ovidiu: Is this a reasonable thing to do?

The only trick is to invalidate these when the propath is assigned. We could store currentTimeMillis() as lookupTimestamp along with each found trigger. If we get a session level notification when propath is assigned, could store currentTimeMillis() as the session-level lastPropathAssignment. When we use the stored trigger program, we just compare the lookupTimestamp to lastPropathAssignment to see if the cached value is still valid.

#35 - 11/23/2023 08:54 AM - Greg Shah

My point here is that by doing something in the database trigger manager itself, we can get an even better level of optimization for minimal effort and risk.

As a side benefit, we leave the (already complicated) SourceNameMapper alone instead of adding trigger-specific logic.

#36 - 11/23/2023 09:03 PM - Ovidiu Maxiniuc

Greg Shah wrote:

> If we implement the solution in #6649, we still have to do a minimal cache lookup for each trigger execution. We can avoid that if the trigger manager would cache the trigger program along with the trigger definition itself.
> Ovidiu: Is this a reasonable thing to do?

As a matter of facts, the DatabaseTriggerManager relies on SourceNameMapper.convertNameToClass(); to obtain the Java class name of the trigger defined in DMO's @Table annotation. It is not cached locally, exactly because we have seen that the trigger may change once the propath is altered.

We know that SourceNameMapper is slow, simply because it MUST traverse the full list of directories for a match. Just by chance, as part of another task, I was just thinking of caching the result of SourceNameMapper.convertNameToClass(); as a way to improve the general performance of the applications, for all procedure calls, not only for triggers. So if the propath is not changed from the last request of a specific external procedure/trigger, the result must be cached (in a map/cache) and returned each time a specific code is needed. When the propath is altered, the cache/map is simply reset.

Constantin, I do not know all details of SourceNameMapper. Is there already such implementation in it? The triggers will surely benefit from it, as will do the calls to the most accessed methods (the 10:90 principle).

#37 - 11/24/2023 12:42 AM - Dănuț Filimon

#6649 already has a cache implementation which not only stores the results from convertNametoClass(), but also convertOOToClass() and lookupLegacyName(). This cache is **not reset** when a new propath is used, but the new propath is used to create a new cache entry based on an existing map of initialization paths that are read from name_map.xml and to which other program names searched are constantly added (in this way, the next cache created will be a lot more complete, but it will also take more time to create - of course, the cache entry is created by a separate thread).

I've looked over Alexandru's (ad) implementation previously and said that there will be no problems with the changes from #6649, now I can slightly see a few cases where these changes will have to be adjusted, for example:

- the convert/lookup methods specified above now rely on fastConvertName which uses a path lookup, then uses convertName if it can't find an answer. The new logic from convertName will need to be moved to fastConvertName;
- One of the recent improvements was to cache the propath hash code in the WorkArea so that it is not computed each time, now the sourceCache will need to use another key made of the propath and trig, and the WorkArea member should be removed.

The following is separate and is related to the implementation from #6496:

- The searchPath is an array with a single value which is sent as a parameter to convertName, and if I understood it correctly, this **trig** path needs to be prepended to each propath. In this case a String array will be created and used, so I don't see the use of sending a String[] with a single parameter will be useful if it's going to be replaced. The current implementation is ok, but after prepending the trig path will not be able to replace the searchPaths so sending a single String as parameter and building the searchPaths from there should be ok.

This will affect the performance slightly since if the cache is not hit, it will need to build the new search paths and do a lookup, in some cases there

are propaths with ~70 entries.

I've done a meld with #6649 and these were the cases where the implementation will have to be changed. In my opinion, the changes will not be hard to make, but will need to be tested accordingly.

#38 - 11/27/2023 08:08 AM - Greg Shah

Ovidiu: Is it difficult for us to cache the results at the trigger definition? If not, I'd prefer that over the idea that we embed trigger knowledge inside SourceNameMapper. Since it will be slightly faster as well, this seems like a double win.

#39 - 11/27/2023 04:50 PM - Ovidiu Maxiniuc

Most likely, not. A context-level map should do it. The only think that is missing is the invalidate event, when the paths are changed.

#40 - 11/28/2023 08:54 AM - Greg Shah

Are the trigger defs already context-specific? If so, let's avoid the map lookup.

#41 - 11/28/2023 04:29 PM - Ovidiu Maxiniuc

The (schema-)triggers are context specific because the propath is context specific (EnvironmentOps.getSearchPath()).
The trigger manager can be registered with EnvironmentOps.addSourcePathListener() and use this event to invalidate the trigger classes. But the heavy lifting will still be done by SourceNameMapper, but only once for each trigger, if the propath is not changed.

#42 - 12/05/2023 06:24 AM - Alexandru Donica

- Status changed from WIP to Review

I have commited an update to branch 6496a rev. 14747.
I saved a cache in DatabaseTriggerManager to get the java class name using the name of the procedure, which is updated every time a new procedure is called. Also, I save an array of strings that represent the search paths, which can either be the trig parameter path if it is an absolute path, otherwise it is the propath with every path concatenated with the trig path. This array is sent to the convertNameToClass method if needed.

And if the propath is used with the trig path, then the DatabaseTriggerManager also registers to EnvironmentOps.addSourcePathListener(). In this case, when the propath changes, the cache is emptied and the saved search paths is also reset.
I have done some small tests and will test some more to see if I missed anything, but a review would also be nice.

#43 - 12/06/2023 06:32 PM - Ovidiu Maxiniuc

Review of 6496a/14743-7:

Good job. I think the code is almost ready. Just a few minor things to adjust.

- DatabaseTriggerManager.java:
  - in history header: usually, we add a single entry per branch merged into trunk, not individual commits. While the evolution of development of a specific feature might be interesting, these notes are available in respective branch log. Indeed, there are some previous entries with notes notes per entry, but that was an exceptional the case when we worked with sub-branches. In this case, only the first note from 20230918 should remain as it summarize the branch/task. Similar for SourceNameMapper.java.
  - please initialize isTrigAbsolutePath when declaring it or, better, make sure it is set for all if-branches in computeTriggerSearchPaths() (the method which computes it). Currently, there are cases when the field remains unchanged (either true (from a previous call) or false (by default)) after the execution of the method.
  - initialization of context member / initialValue(): the call to get() is time-consuming. Please cache its value into a local variable.
- SourceNameMapper.java
  - even if they share the same name, the convertNameToClass() methods must be separated by an empty line;
  - lines 2125 and 2160: space is missing between if keyword and (;
  - line 2114 is too long. Chop the list of parameters to make the method fit into line limit.

I cannot pretend I have fully understood all the changes in SourceNameMapper. Overall, the code in each method seems fine, and the fact that some

optional parameters were added should not affect the code. However, because of the overloaded methods it's difficult to track the execution so I am relying on javadocs and local comments.

After addressing the above issue please rebase the branch to latest trunk.

#44 - 12/06/2023 08:57 PM - Ovidiu Maxiniuc

Had an idea for local optimisation: in DTM.getSchemaTrigger(), proceduresClassMap is used to cache the converted procedures names. That is OK, the calls to SourceNameMapper for obtaining the Java counterpart of progress procedures is avoided in subsequent fires of same trigger. I was thinking whether we could map the Class instead, so we avoid it being mapped/ loaded and validated (to extend DatabaseTrigger) each time.

#45 - 12/07/2023 04:08 AM - Alexandru Donica

Commited the changes requested in branch 6496a rev 14748.
Also, should I add more comments inside the methods to explain the flow, or is the code sufficiently self-explanatory/logically structured?

The rebase is yet to be done.

#46 - 12/07/2023 06:28 AM - Alexandru Lungu

Rebased 6496a to latest trunk. 6496a is now at rev. 14869
Alexandru [ad], please review StartupParameters as there was some non-trivial rebasing process there. I moved the trigger location parameter handling to the unified routines we have now. Check that it is still properly picking up the trigger location from the start-up parameters.

#47 - 01/19/2024 01:13 AM - Eric Faulhaber

Alexandru [either one ;)]: what is left to do to push this task over the finish line?

#48 - 01/19/2024 02:11 AM - Alexandru Lungu

- Status changed from Review to Internal Test

Alexandru, please test the large application with this parameter you introduced:

- check there are no regressions with regression testing
- check there are no performance regressions
- debug to check that your changes are picked up and work properly

Lets get this done today.

#49 - 01/19/2024 09:53 AM - Alexandru Donica

I made a copy of 7156b and put my changes there, merged the changes to SourceNameMapper that interfered with my changes, tested them and the changes to StartupParameters with a small test of mine, and have only to test with the larger apps/tests.

#50 - 01/23/2024 09:32 AM - Alexandru Donica

I tested large app and performance tests with and without changes. The performance tests did not seem to have many triggers called (I placed a break point which was hit only during the warm-up), so I tried to count the number of paths checked when looking for a trigger class, and the total time it takes to search for all searches. I started some functional tests for the large app, and the number of paths that were checked for the branch with the changes were about the same (slightly better) than the no. of paths checked for the base branch, but the total time to find the trigger class was smaller (500ms vs 700ms), but this could be greatly influenced by the -trig parameter itself (whether it's more specific, or just a "triggers" folder, or if the class is not found, etc.). In some cases the time can be worse, depending on the trig path and propath.

Other than that, there was no performance decrease observable, the same amount of tests failed for this branch as for the base branch, and I am sure that the change is picked up.
Should I commit the changes from a branch copied from trunk, or from 7156b (which I used to test)?

#51 - 01/23/2024 10:43 AM - Alexandru Lungu

Alexandru [ad], 7156b is mostly like a fast-paced trunk. We periodically rebase it. We only commit hot patches there.

These changes can reach trunk and be picked up later by 7156b. I created 6496b. 6496a is too far behind and is facing some hard rebase challenges with SourceNameMapper.

#52 - 01/25/2024 04:06 AM - Alexandru Donica

- Status changed from Internal Test to Review

Commited to branch 6496b rev 14943.

#53 - 01/30/2024 03:50 PM - Eric Faulhaber

Alexandru Donica wrote:

> Commited to branch 6496b rev 14943.

Ovidiu, please review, at least the trigger implementation changes. We are trying to get this into trunk ASAP.

Can someone familiar with SourceNameMapper please review the changes to that class?

I'm not the best person to review this change set, but I took a quick look. How expensive is the new initialization going on for every context in DatabaseTriggerManager.context? Alexandru [ad], you noted there was no notable performance degradation. Were you considering a cold server or warm server? We have a "first iteration of an operation is too slow" problem in general. Just looking at this from the perspective of the one-time hit when a new context is established, is this likely to make it worse, or is this processing pretty negligible?

#54 - 01/30/2024 05:59 PM - Ovidiu Maxiniuc

Review of 6496b rev 14943.

- the copyright year must be updated (and the branch rebased) in each altered file;
- DatabaseTriggerManager.java
  - line 270: if the computation ended, we should set triggerSearchPaths to some value (possibly new String[0]) to mark that it was already processed. Otherwise, the method computeTriggerSearchPaths() will be invoked multiple times (the condition for this is triggerSearchPaths being null) with same outcome, since the triggerSearchPaths remains null after each invocation.
    LE: I saw this checked at line 384. In the current form, the triggerSearchPaths cannot be set at this point. Anyway, if I read the code correctly, triggerSearchPaths remains null after the call to computeTriggerSearchPaths() only if the -trig parameter is an absolute path but invalid, in which case the standard (trigger-) procedure lookup takes place;
  - line 288: the line is superfluous. This is the first line of the else of if where isTrigAbsolutePath is checked for true;
  - lines 289-295: question: in case of absolute path, the project token is tested/used. It may be debatable, but shouldn't we take into consideration the token when the -trig path is not absolute? This is FWD specific, of course.
  - line 410: local.proceduresClassMap.put(procedure, cls); should be enough. We have already decided that the map does not contain the procedure key (the reason for entering the if at line 351)
- SourceNameMapper.java

- line 2617: usage of enhanced for over an array. The implementation uses iterators which are slower than direct array access.
- the rest seem fine to me, even if I am not the best suited reviewer here.

#55 - 01/31/2024 04:38 AM - Alexandru Lungu

Alexandru [ad], please address the review. We need to get this merged asap.

#56 - 01/31/2024 06:32 AM - Greg Shah

Hynek: Please review the SourceNameMapper changes.

#57 - 01/31/2024 06:34 AM - Alexandru Donica

Ovidiu Maxiniuc wrote:

> Review of 6496b rev 14943.
>
> - DatabaseTriggerManager.java
>   - line 270: if the computation ended, we should set triggerSearchPaths to some value (possibly new String[0]) to mark that it was already processed. Otherwise, the method computeTriggerSearchPaths() will be invoked multiple times (the condition for this is triggerSearchPaths being null) with same outcome, since the triggerSearchPaths remains null after each invocation.
>   LE: I saw this checked at line 384. In the current form, the triggerSearchPaths cannot be set at this point. Anyway, if I read the code correctly, triggerSearchPaths remains null after the call to computeTriggerSearchPaths() only if the -trig parameter is an absolute path but invalid, in which case the standard (trigger-) procedure lookup takes place;

I modified this part, changed trigPath to an Optional<String> to easily check if we never looked for it yet or if we did and did/did not find it. If we don't find it it is pointless to try to compute the trig paths. Now there should be no scenario where it tries to compute again and again and still result in an empty list.

- lines 289-295: question: in case of absolute path, the project token is tested/used. It may be debatable, but shouldn't we take into consideration the token when the -trig path is not absolute? This is FWD specific, of course.

For the part with the token I took inspiration from SourceNameMapper, where it was used only for absolute paths when searching for a class, not relative paths. I hope it is correct. This is the comment in SourceNameMapper explaining the reasoning (line 2806):

```
// one more try, because of file collisions, the files from current project were
// converted from a 'token' directory of the src-root
```

Commited all the changes you mentioned to branch 6496b rev. 14944.

#58 - 01/31/2024 09:58 AM - Hynek Cihlar

Code review 6496b. The SourceNameMapper changes look good. Just please update the copyright year in StartupParameters.java and remove the only single change in StartupParameter.java.

#59 - 01/31/2024 11:20 AM - Alexandru Donica

Sorry for mistaking the files.
Commited change to branch 6496b rev 14945.

#60 - 02/01/2024 09:26 AM - Greg Shah

- Status changed from Review to Internal Test

What testing remains?

#61 - 02/01/2024 05:03 PM - Alexandru Lungu

AFAIK, these were tested on a large customer application regression tests and POC by actually setting the flag and debugging into it to see if it picks it up - **and it does**.

This is not risky is the sense that it doesn't spoil normal executions. Only if the setting is set, the new code is used - which hits the more "risky part".

From my understanding, this can be merged to trunk.

#62 - 02/02/2024 09:17 AM - Constantin Asofiei

- Status changed from Internal Test to Merge Pending

#63 - 02/02/2024 11:36 AM - Alexandru Lungu

- Status changed from Merge Pending to Test

Branch 6496b was merged into trunk as rev 14965 and archived.