Database - Bug #7467

join on meta table

06/25/2023 06:53 AM - Constantin Asofiei

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

Description

History

#1 - 06/25/2023 06:58 AM - Constantin Asofiei

This test:

```
find first _index-field.
find _index of _index-field no-lock.
```

shows this during conversion:

```
com.goldencode.p2j.schema.SchemaException: Invalid join: fwd._index of fwd._index-field
    at com.goldencode.p2j.schema.RelationAnalyzer.createJoin(RelationAnalyzer.java:309)
    at com.goldencode.p2j.schema.SchemaWorker$SchemaSupport.recordJoin(SchemaWorker.java:846)
    at com.goldencode.expr.CE3138.execute(Unknown Source)
    at com.goldencode.expr.Expression.execute(Expression.java:398)
    at com.goldencode.p2j.pattern.Rule.apply(Rule.java:500)
    at com.goldencode.p2j.pattern.Rule.executeActions(Rule.java:751)
    at com.goldencode.p2j.pattern.Rule.coreProcessing(Rule.java:717)
    at com.goldencode.p2j.pattern.Rule.apply(Rule.java:537)
    at com.goldencode.p2j.pattern.Rule.executeActions(Rule.java:751)
    at com.goldencode.p2j.pattern.Rule.coreProcessing(Rule.java:717)
    at com.goldencode.p2j.pattern.Rule.apply(Rule.java:537)
    at com.goldencode.p2j.pattern.RuleContainer.apply(RuleContainer.java:593)
    at com.goldencode.p2j.pattern.RuleSet.apply(RuleSet.java:1)
    at com.goldencode.p2j.pattern.AstWalker.walk(AstWalker.java:262)
    at com.goldencode.p2j.pattern.AstWalker.walk(AstWalker.java:210)
    at com.goldencode.p2j.pattern.PatternEngine.apply(PatternEngine.java:1701)
    at com.goldencode.p2j.pattern.PatternEngine.processAst(PatternEngine.java:1579)
    at com.goldencode.p2j.pattern.PatternEngine.processAst(PatternEngine.java:1510)
    at com.goldencode.p2j.pattern.PatternEngine.run(PatternEngine.java:1062)
    at com.goldencode.p2j.convert.TransformDriver.processTrees(TransformDriver.java:589)
    at com.goldencode.p2j.convert.ConversionDriver.middle(ConversionDriver.java:391)
    at com.goldencode.p2j.convert.TransformDriver.executeJob(TransformDriver.java:995)
    at com.goldencode.p2j.convert.ConversionDriver.main(ConversionDriver.java:1284)
```

and at runtime shows this:

```
** meta_index of meta_index_field. Index fields of table 1 must be fields in table 2. (230)
```

The conversion is like this:

```
         new FindQuery(metaIndexField, (String) null, null, "metaIndexField.indexRecid asc, metaIndexField.ind
exSeq asc").first();
```

```
         new FindQuery(metaIndex, (String) null, null, "metaIndex.fileRecid asc, metaIndex.indexName asc", met
aIndexField, LockType.NONE).unique();
```

Is this a problem with standard.df ?

Is this a conversion problem? Can this be solved only at runtime?

#3 - 06/25/2023 07:00 AM - Constantin Asofiei

The customer's application has these errors:

```
     [java] com.goldencode.p2j.schema.SchemaException: Invalid join: fwd._file of fwd._field
     [java] com.goldencode.p2j.schema.SchemaException: Invalid join: fwd._field of fwd._index-field
     [java] com.goldencode.p2j.schema.SchemaException: Invalid join: fwd._index of fwd._index-field
```

so this is not just about the example above, it seems to be a more general problem.

#4 - 06/26/2023 01:58 PM - Ovidiu Maxiniuc

This seems like a problem with finding the inverse join index. In the case of these tables, the definitions are similar to these:

```
DEFINE TABLE _Index
   FIELD _File-recid AS RECID MAX-WIDTH 8 INITIAL ? LABEL File-recid
   FIELD _Index-Name AS CHARACTER MAX-WIDTH 32 INITIAL ? LABEL Index-Name
   FIELD _Unique AS LOGICAL MAX-WIDTH 1 INITIAL true LABEL Unique
   FIELD _num-comp AS INTEGER MAX-WIDTH 4 INITIAL 0
   FIELD _idx-num AS INTEGER MAX-WIDTH 4 INITIAL ? LABEL idx-num
   FIELD _I-misc1 AS INTEGER EXTENT 8 MAX-WIDTH 80 INITIAL ?
   FIELD _I-misc2 AS CHARACTER EXTENT 8 MAX-WIDTH 208 INITIAL ?
   FIELD _Active AS LOGICAL MAX-WIDTH 1 INITIAL true LABEL Active
   FIELD _I-res1 AS INTEGER EXTENT 8 MAX-WIDTH 80 INITIAL ?
   FIELD _I-res2 AS CHARACTER EXTENT 8 MAX-WIDTH 208 INITIAL ?
   FIELD _Wordidx AS INTEGER MAX-WIDTH 4 INITIAL ?
   FIELD _Desc AS CHARACTER CASE-SENSITIVE MAX-WIDTH 4096 INITIAL ?
   FIELD _For-Name AS CHARACTER CASE-SENSITIVE MAX-WIDTH 64 INITIAL ?
   FIELD _For-Type AS CHARACTER CASE-SENSITIVE MAX-WIDTH 24 INITIAL ?
   FIELD _ianum AS INTEGER MAX-WIDTH 4 INITIAL ? LABEL ianum
   FIELD _Idx-CRC AS INTEGER MAX-WIDTH 4 INITIAL ?
   FIELD _Idxowner AS CHARACTER MAX-WIDTH 32 INITIAL PUB
   FIELD _Idxmethod AS CHARACTER MAX-WIDTH 2 INITIAL ?
      FIELD _User-Misc AS CHARACTER MAX-WIDTH 20 INITIAL ? LABEL User-Misc
      FIELD _Last-modified AS DATETIME-TZ MAX-WIDTH 12 INITIAL ?
      FIELD _Mod-sequence AS INTEGER MAX-WIDTH 4 INITIAL ?
   FIELD _Mode AS INTEGER MAX-WIDTH 4 INITIAL 0 LABEL Mode
   FIELD _Word-Rules AS INTEGER MAX-WIDTH 4 INITIAL ? LABEL Word-Rules
   FIELD _Index-Attributes AS LOGICAL EXTENT 64 MAX-WIDTH 512 INITIAL false LABEL Index-Attributes

   INDEX _File/Index AS PRIMARY UNIQUE _File-recid _Index-Name
```

```
   INDEX _Index-Name AS _Index-Name
   INDEX _Index-Number AS _idx-num
```

and respectively:

```
DEFINE TABLE _Index-Field
   FIELD _Index-recid AS RECID MAX-WIDTH 8 INITIAL ? LABEL Index-recid
   FIELD _Index-Seq AS INTEGER MAX-WIDTH 4 INITIAL ? LABEL Index-Seq
   FIELD _Field-recid AS RECID MAX-WIDTH 8 INITIAL ? LABEL Field-recid
   FIELD _Ascending AS LOGICAL MAX-WIDTH 1 INITIAL true LABEL Ascending
   FIELD _Abbreviate AS LOGICAL MAX-WIDTH 1 INITIAL false LABEL Abbreviate
   FIELD _Unsorted AS LOGICAL MAX-WIDTH 1 INITIAL false LABEL Unsorted
   FIELD _If-misc1 AS INTEGER EXTENT 8 MAX-WIDTH 80 INITIAL ?
   FIELD _If-misc2 AS CHARACTER EXTENT 8 MAX-WIDTH 208 INITIAL ?
   FIELD _If-res1 AS INTEGER EXTENT 8 MAX-WIDTH 80 INITIAL ?
   FIELD _If-res2 AS CHARACTER EXTENT 8 MAX-WIDTH 208 INITIAL ?
      FIELD _User-Misc AS CHARACTER MAX-WIDTH 20 INITIAL ? LABEL User-Misc
      FIELD _Last-modified AS DATETIME-TZ MAX-WIDTH 12 INITIAL ?
      FIELD _Mod-sequence AS INTEGER MAX-WIDTH 4 INITIAL ?
   FIELD _Collation AS CHARACTER MAX-WIDTH 38 LABEL Collation
   FIELD _Coll-Strength AS INTEGER MAX-WIDTH 4 INITIAL 0
   FIELD _KeyValue-expr AS CHARACTER CASE-SENSITIVE MAX-WIDTH 4096 LABEL KeyValue-expr
   FIELD _If-Attributes AS LOGICAL EXTENT 64 MAX-WIDTH 512 INITIAL ? LABEL If-Attributes

   INDEX _Index/Number AS PRIMARY UNIQUE _Index-recid _Index-Seq
   INDEX _Field AS _Field-recid
```

The expected navigation join is using _Index-Field._Index-recid = RECID(_Index). Right?
However, our algorithm (TableMapper.findJoiningIndex()) attempts to locate the common fields (and correctly identify them as: _User-Misc, _Last-modified, _Mod-sequence - I indented them to make more visible) but these are not related and/therefore they do not form any index.

#5 - 06/26/2023 02:22 PM - Constantin Asofiei

Ovidiu, this looks troublesome. 4GL may have some even more hidden fields to be able to join the meta tables, or some hard-coded rules when using FIND ... OF ... for a parent/child meta table relation.

I've tried making some equivalent temp-tables for the _Index and _Index-field definitions you posted, and I can't make the FIND ... OF work.

I think we need to find how we can 'break' the FIND ... OF inverse relation. USE-INDEX seems to be ignored.

The conclusion may be that the relation is hardcoded via the _<parent-table>-recid field in the child table, for the meta tables. The good news is we may hard-code this at runtime.

#6 - 06/26/2023 09:05 PM - Ovidiu Maxiniuc

- Status changed from New to WIP

Constantin, please try the following patch:

```
--- a/src/com/goldencode/p2j/persist/AbstractJoin.java
+++ b/src/com/goldencode/p2j/persist/AbstractJoin.java
@@ -148,12 +148,15 @@
       this.inverseIface = inverse.getDMOInterface();
       this.inverse = inverse;

-      RelationInfo info = local.getDmoInfo().getRelation(inverseIface);
+      DmoMeta inverseInfo = inverse.getDmoInfo();
+      DmoMeta localInfo = local.getDmoInfo();
+      RelationInfo info = localInfo.getRelation(inverseIface);
       boolean dereference = (info != null);
+      String parentChildRelationField = null;

       if (!dereference)
       {
-         info = inverse.getDmoInfo().getRelation(localIface);
+         info = inverseInfo.getRelation(localIface);
       }

       // if relation doesn't exist, add it at runtime
@@ -198,14 +201,26 @@
             }
             else if (inverseIndex == null)
             {
-               raiseJoinError(local, inverse);
+               // "FIND <parent> OF <child>", based on child having a "FIELD <parent>-recid AS RECID" field
+               // [local] is <parent> and [inverse] is <child>
+               Property property = inverseInfo.byLegacyName(localInfo.legacyTable.toLowerCase() + "-recid");
+               if (property == null)
+               {
+                  raiseJoinError(local, inverse);
+               }
+               else
+               {
+                  parentChildRelationField = property.name;
+               }
             }

             RecordBuffer referring = localHasLeadingIndex ? inverse : local;
             RecordBuffer referent = localHasLeadingIndex ? local : inverse;
             P2JIndex relationIndex = localHasLeadingIndex ? localIndex : inverseIndex;

-            Set<String> indexFieldNames = TableMapper.getIndexFieldNames(relationIndex);
+            Set<String> indexFieldNames = relationIndex == null
+                  ? new HashSet<>(Collections.singletonList(parentChildRelationField))
+                  : TableMapper.getIndexFieldNames(relationIndex);
             info = new RelationInfo(referring.getDMOInterface(),
                                     referring.getDMOImplementationClass(),
                                     referent.getDMOInterface(),
@@ -221,11 +236,11 @@
             {
                fieldName = fieldName.toLowerCase();
                String referringProp = legacyToPropReferring.get(fieldName);
-               String referentProp  = legacyToPropReferent.get(fieldName);
+               String referentProp = legacyToPropReferent.get(fieldName);
                Boolean ignoreCase = referent.getDmoInfo().isIndexedIgnoreCase(referentProp);
                info.addPropertyData(referringProp,
                                     referentProp,
-                                    ignoreCase != null? ignoreCase.booleanValue() : false);
+                                    ignoreCase != null ? ignoreCase.booleanValue() : false);
             }

             dereference = !localHasLeadingIndex;
--- a/src/com/goldencode/p2j/persist/DBUtils.java
```

```
+++ b/src/com/goldencode/p2j/persist/DBUtils.java
@@ -1168,6 +1168,12 @@
     */
    static void composePropertyName(RecordBuffer buffer, String name, StringBuilder buf, Database.Type type)
    {
+      if (name == null)
+      {
+         buf.append(Session.PK);
+         return;
+      }
+
       boolean isChar = false;
       boolean ignoreCase = false;
       String computedColumnPrefix = null;
```

It fixed the testcase for me and the result is similar to what I see in 4GL.

However, this will only allow you to advance with your runtime. I think the correct solution involves the partial usage of _Index/Number index of _Index-Field. If you look at this index, it contains both _Index-recid and _Index-Seq. The first field is clear what it means, but the second is still a puzzle, at least for me. OTOH, I was expecting the query to use an index of the 'parent' table to quickly lookup the _Index record OF the specified _Index-Field.

#7 - 06/27/2023 02:56 AM - Constantin Asofiei

Ovidiu, _Index-Seq I think is the position of the field in the index definition. Also, we need to limit this fix only for meta tables. I don't have a customer app standalone test for this issue yet.

#8 - 06/27/2023 01:44 PM - Ovidiu Maxiniuc

Constantin Asofiei wrote:

> Ovidiu, _Index-Seq I think is the position of the field in the index definition.

Actually, I do not think it does matter here. The _Index/Number index is used to iterate records from _Index-Field. We are looking up records from _Index instead, and, most likely, this is done by direct access, using the record's PK. As we learnt recently, in 4GL, this is even faster than using an index since the recid offers the exact offset of the record in the file.

> Also, we need to limit this fix only for meta tables.

Yes, this can be done. I also tried to duplicate this with temp-tables and failed to duplicate, I will also attempt to do this with permanent tables.

> I don't have a customer app standalone test for this issue yet.

I thought you encountered this during the testing of the new revision of customer application.

#9 - 06/27/2023 02:01 PM - Constantin Asofiei

Ovidiu Maxiniuc wrote:

> > I don't have a customer app standalone test for this issue yet.
>
> I thought you encountered this during the testing of the new revision of customer application.

The issue was noticed during conversion (the exception for the ~~index~~ join), and the standalone test was determined from looking at the customer's source code.

#10 - 06/27/2023 04:58 PM - Ovidiu Maxiniuc

Constantin Asofiei wrote:

> Ovidiu, _Index-Seq I think is the position of the field in the index definition. Also, we need to limit this fix only for meta tables. I don't have a customer app standalone test for this issue yet.

I will prepare a new branch (7467a) starting from the patch above, with this filter applied.

> The issue was noticed during conversion (the exception for the ~~index~~ join), [...].

Sorry, because this is not fatal and the procedure is still converted to a Java class I missed the Invalid join: notification.
This was partially due to multitude of Could not find legacy builtin method javaname for [...] messages in the conversion log (but this is not the location for this discussion).

#11 - 06/27/2023 08:39 PM - Ovidiu Maxiniuc

I did some more research.

Indeed, in the case of TEMP-TABLES this <parent>-recid lookup does not work. Here is the example:

```
define temp-table parent
    field f1 as char.

define temp-table child
    field parent-recid as recid
    field parent-seq as integer
    field f2 as char
    index abcdx as primary unique parent-recid parent-seq.

create parent. f1 = "f".
create child. parent-recid = recid(parent). f2 = "g". parent-seq = 1.
release parent.
release child.

find first child.
find first parent of child.
message f1 f2.
```

This piece of code will refuse to compile, the errors are:

```
** "parent of child".  Index fields of table 1 must be fields in table 2. (230)
** <program> Could not understand line 19. (196)
```

However, in the case of permanent tables, things are different. I created similar tables in my fwd test database:

```
ADD TABLE "Child"
  DUMP-NAME "child"

ADD FIELD "parent-recid" OF "Child" AS recid
  FORMAT ">>>>>>9"
  POSITION 2
  ORDER 10

ADD FIELD "parent-seq" OF "Child" AS integer
  FORMAT "->,>>>,>>9"
  POSITION 3
  ORDER 20

ADD FIELD "f2" OF "Child" AS character
  FORMAT "x(8)"
  POSITION 4
  ORDER 30

ADD INDEX "abcdx" ON "Child"
  UNIQUE
  PRIMARY
  INDEX-FIELD "parent-recid" ASCENDING
  INDEX-FIELD "parent-seq" ASCENDING

ADD TABLE "Parent"
  DUMP-NAME "parent"

ADD FIELD "f1" OF "Parent" AS character
  FORMAT "x(8)"
  POSITION 2
  MAX-WIDTH 16
  ORDER 10
```

and executed the same procedure (with temp-table definitions removed):

```
create parent. f1 = "f".
create child. parent-recid = recid(parent). f2 = "g". parent-seq = 1.
release parent.
release child.

find first child.
find first parent of child.
message f1 f2.
```

The output is

```
f g
```

The index is ignored. I dropped it (actually I had to recreate the child table without any indices) and the result is exactly the same.

#12 - 06/28/2023 04:15 AM - Constantin Asofiei

Ovidiu, so this means that this 'recid relation' is automatically applied only to meta or permanent tables. That is good news, at least we have some consistency.

But, please test what happens if you remove the child.parent-recid field - does find first parent of child. still work?

#13 - 06/30/2023 10:04 PM - Ovidiu Maxiniuc

Constantin Asofiei wrote:

> Ovidiu, so this means that this 'recid relation' is automatically applied only to meta or permanent tables. That is good news, at least we have some consistency.

Yes. They are.

> But, please test what happens if you remove the child.parent-recid field - does find first parent of child. still work?

I did extensive tests between tables from permanent, _meta, and _temp databases. They proved the relation can exist only on _meta and permanent databases. (Only) When fields named according to the above schema exists the joins are possible.

Cross databases joins using this PARENT-ID relation is not possible. The _meta excludes itself automatically since we cannot create tables/fields starting with underscore so the join field is inexistent. When a _temp table is involved I identified two kind of errors, but both of them are compile-time. Similarly, FWD will report the issues at conversion time, but will allow the process to continue, delegating the (partial) errors at runtime.

Please find my changes in 7467a, revision 14636.

#14 - 06/30/2023 10:05 PM - Ovidiu Maxiniuc

- Assignee set to Ovidiu Maxiniuc

- % Done changed from 0 to 100

- Status changed from WIP to Review

#15 - 07/05/2023 01:16 PM - Constantin Asofiei

Ovidiu, we need this in trunk soon. What testing is needed?

Also, do you still see the com.goldencode.p2j.schema.SchemaException: Invalid join during conversion?

Eric: I think is better for you to review it.

#16 - 07/05/2023 03:13 PM - Ovidiu Maxiniuc

Constantin Asofiei wrote:

> Ovidiu, we need this in trunk soon. What testing is needed?

The conversion does not actually generate different output, just the log is changed. In trunk, the PARENT-RECID relation is not recognized so the attempt to use this kind of JOIN will print a SchemaException: Invalid join for the respective tables. With 7467a, these are now recognized and reported as normal joins in ALL NATURAL JOINS: section. I tested the conversion only for really small applications: a **full conversion** of customer application is recommanded, just to be sure I did not introduced involuntarily any regression.
The same situation we have for the runtime. In the event no classic, index-based joins are identified and neither of the tables are temporary, the test for PARENT-RECID join is done. Based on the initial notes of this task you should be able to see whether the original errors/warnings are gone only by **updating the runtime**.

> Also, do you still see the com.goldencode.p2j.schema.SchemaException: Invalid join during conversion?

Yes, for tables which cannot be joined using any of the three known relations (many-to-one, one-to-one and the new parent-pk).

#17 - 07/05/2023 06:31 PM - Eric Faulhaber

Code review 7467a/14636:

Sorry, it's been a long time since I looked at this code and it has changed quite a lot from what I remember. Overall, the changes are most likely fine, but I have a few questions:

- Why do we return Session.PK when the name parameter is null in DBUtils.composePropertyName?
- I don't fully follow the bestCandidate algorithm in RelationAnalyzer.attemptParentIdJoin. Can you please provide a practical example where there are multiple joins/fields which could be chosen and how this algorithm chooses the best candidate? What makes one candidate the best choice?
- I don't recall how SchemaWorker.getForeignKey is used; why do we return null (or possibly empty string, which is commented out with a

question mark) as the foreign key for the parent relation join type?
- Have you tested these changes with dynamic queries using the <table A> OF <table B> semantic?

#18 - 07/05/2023 09:34 PM - Ovidiu Maxiniuc

Eric Faulhaber wrote:

> Code review 7467a/14636:
>
> Sorry, it's been a long time since I looked at this code and it has changed quite a lot from what I remember. Overall, the changes are most likely fine, but I have a few questions:
>
> - Why do we return Session.PK when the name parameter is null in DBUtils.composePropertyName?

This is the way I identify the PARENT-RECID relation between the joined tables. Up until now, having null as parameter here was not possible except for a FWD flaw. My bad, I should have added the necessary javadoc here. Actually, null is quite a good constant, otherwise we will need to add a reserved literal for PK.

> - I don't fully follow the bestCandidate algorithm in RelationAnalyzer.attemptParentIdJoin. Can you please provide a practical example where there are multiple joins/fields which could be chosen and how this algorithm chooses the best candidate? What makes one candidate the best choice?

From my tests, the algorithm for matching the 'relation' field is more complex. If there is no perfect match like <parent>-recid, 4GL will use any RECID field named <parent>-recid<suffix>. I guess they only compare the first letters of the identifiers. I noticed this when Constantin asked to remove the field and see whether the join is still possible. Instead, I renamed it to <parent>-recid2 and I had the surprise that the join was still possible. If another, stricter-named field is add, that one will be used instead.

> - I don't recall how SchemaWorker.getForeignKey is used; why do we return null (or possibly empty string, which is commented out with a question mark) as the foreign key for the parent relation join type?

This is only used at conversion. For the special case of PARENT_ID_RELATION join, the of_foreign_key is not saved, it is implied. My initial solution was to use an empty string, but this was only complicating the AST.

> - Have you tested these changes with dynamic queries using the <table A> OF <table B> semantic?

No. Thank you for the idea. I tested now and my testcases work for 80% of cases. I am getting some strange Namespace.findUnambiguously() errors, but I need further investigations here because I have several other patches applied to my working FWD.

I will add some more javadoc with explanations of the decisions I picked and shortly descried above. I should have done a review of my own code after checking it works as I expected.

#19 - 07/06/2023 08:42 AM - Constantin Asofiei

Ovidiu, is the relation info added at the DMOs during conversion mandatory for the runtime to work properly? I ask because during incremental conversion, this info can get lost. Please test what happens at runtime if you manually remove the relation (related to the join) from the DMO annotations.

#20 - 07/06/2023 09:05 PM - Ovidiu Maxiniuc

Unlike the other two join relations, this is not saved to DMO interface. The runtime code which analyses the involved DMOs tries to pick one of those in first instance. If none is found, it checks whether there is a field in the left-side table which matches <right-side-table>-recid<optional-prefix> pattern. This is very quick (a single map lookup) compared to the other join relations where the index fields need to be iterated.

So you do not need to worry about the incremental conversion here. The static conversion does only a preliminary check and report the situations where 4GL raises errors.

#21 - 07/10/2023 09:48 AM - Constantin Asofiei

What about relations which do get saved at the DMO interface? Are these mandatory for the FIND ... OF ... to work properly? Because during incremental conversion, the DMOs are generated fully, and if the program which had this relation is not included, this info will get lost.

#22 - 07/11/2023 05:02 PM - Ovidiu Maxiniuc

If a relation is not detected at conversion time or if the annotation was removed, the persistence will always attempt to detect the relation between the two tables at runtime, like for parent-recid relation. After this effort, the new dynamic relation is saved for later queries.

There seems to be some issues with the foreign-key mode (enabled by persistence/foreign-keys in the directory, which forces DynamicJoin as opposed to DynamicLegacyKeyJoin) but this is not the default. Nevertheless, this is an old feature, since before #4011 (I could not find the redmine tracker for it) so it either should restore support for this kind of joins, or drop it.

I rebased the 7467a branch for easier review.

Constantin, is it OK for you to run a quick ETF test on this? Conversion is not mandatory, but since some of the affected classes are used in conversion it is the safer way. Thank you!

#23 - 07/12/2023 01:28 PM - Constantin Asofiei

Ovidiu Maxiniuc wrote:

> Constantin, is it OK for you to run a quick ETF test on this? Conversion is not mandatory, but since some of the affected classes are used in conversion it is the safer way. Thank you!

Conversion is OK - no changes. Runtime testing passed - but there are regressions in trunk. It may be because of fwd-h2. I did not investigate further.

#24 - 07/17/2023 02:25 AM - Eric Faulhaber

Constantin Asofiei wrote:

> Ovidiu Maxiniuc wrote:
>
> > Constantin, is it OK for you to run a quick ETF test on this? Conversion is not mandatory, but since some of the affected classes are used in conversion it is the safer way. Thank you!
>
> Conversion is OK - no changes. Runtime testing passed - but there are regressions in trunk. It may be because of fwd-h2. I did not investigate further.

Is there any reason to keep this task in review status, if these regressions are not related to this task? Is there anything left to be done for this issue?

#25 - 07/17/2023 10:51 AM - Constantin Asofiei

Eric Faulhaber wrote:

> Is there any reason to keep this task in review status, if these regressions are not related to this task? Is there anything left to be done for this issue?

IMO it can be merged to trunk.

#26 - 07/17/2023 11:21 AM - Greg Shah

Please go ahead with the merge, after 6976a.

#27 - 07/17/2023 12:26 PM - Ovidiu Maxiniuc

- Status changed from Review to Test

7467a was merged into trunk as r14658. The branch was archived.