Database - Support #6709

track nested FIND inside a related FOR loop

08/25/2022 11:49 AM - Greg Shah

Status:	WIP	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Dănuț Filimon	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

Related issues:

Related to Database - Support #7004: eliminate the off-end processing in the ...	**New**

History

#1 - 08/25/2022 11:54 AM - Greg Shah

We have the perception that we have a problem with nested FINDs that reference a buffer or buffers controlled by a containing FOR loop. We should track how often this happens and if it is really as prevalent as we think, then we should measure the alternative joined approach to see what we could expect from such an optimization. This may be a first step in deciding the importance of #3194.

#3 - 10/26/2022 06:04 AM - Alexandru Lungu

- Assignee set to Dănuț Filimon

#4 - 11/16/2022 05:36 AM - Dănuț Filimon

From what I understood, we want to test if either nested find or joined approach is better. After a few tests using two temporary tables, tt1 and tt2 with the same structure, I found out the following cases:

1. If the record in tt2 is not found, the code inside the for loop will not be executed. An error is thrown for failing to find a record.

```
FOR EACH tt1.
   FIND FIRST tt2 WHERE tt2.field-tt21 = tt1.field-tt11.
   tt1.field-tt11 = RANDOM(1, 10) MODULO 10.
END.
```

2. If the record in tt2 is not found, the code inside the for loop is still executed because NO-ERROR is used.

```
FOR EACH tt1.
   FIND FIRST tt2 WHERE tt2.field-tt21 = tt1.field-tt11 NO-ERROR.
   tt1.field-tt11 = RANDOM(1, 10) MODULO 10.
END.
```

3. The code is executed only if a record is found.

```
FOR EACH tt1, FIRST tt2 WHERE tt2.field-tt21 = tt1.field-tt11.
   tt1.field-tt11 = RANDOM(1, 10) MODULO 10.
END.
```

There is a big difference between using nested finds and join. Case **3** is similar to case **1**, but there are instances where the user wants to run code when a record doesn't exist (case **2**). I want to know if I addressed the problem correctly and if I should continue to research the issue. I don't think that it will take much long if what I found until now proves helpful.

#5 - 11/16/2022 05:57 AM - Dănuț Filimon

- Status changed from New to WIP

#6 - 11/16/2022 06:34 AM - Greg Shah

This is a good analysis. It certainly highlights the differences in control flow and danger of refactoring such code.

However, the core questions for this task are:

- We are trying to evaluate how often in real 4GL code does this condition occur? Getting a measurement of this for different applications will help us evaluate the importance of optimizations in this area. As your findings above prove, such optimizations would have to be done carefully. Without some measurement of how often this occurs, we are just speculating that this "nested related FIND in a FOR" is a common problem.
- If this is as common as we think, then we would like to know if refactoring of the nested FIND as a multi-table FOR could be used to consistently improve performance.

#7 - 11/18/2022 05:51 AM - Dănuț Filimon

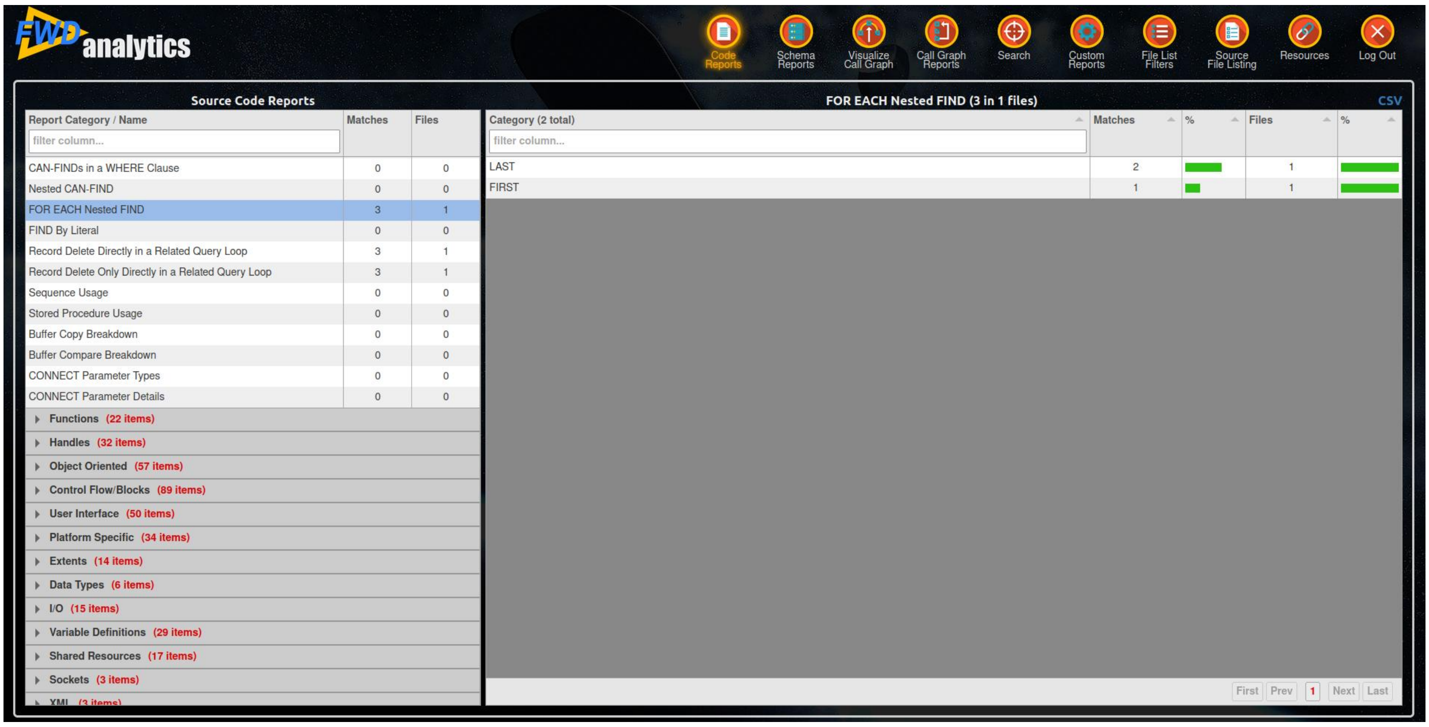
- File report-nested-find.png added

Greg Shah wrote:

> We are trying to evaluate how often in real 4GL code does this condition occur? Getting a measurement of this for different applications will help us evaluate the importance of optimizations in this area. As your findings above prove, such optimizations would have to be done carefully. Without some measurement of how often this occurs, we are just speculating that this "nested related FIND in a FOR" is a common problem.

Thank you for your explanation.

I created a new report for tracking nested FIND occurrences. After running the ant rpt command, we can check how many times this type of operation appeared in the report interface. I decided to track both **nested FIND FIRST** and **FIND LAST** and display them as categories, it will also show the percentage for each one which is very useful. Here is a screenshot of a simple 4GL program with 2 nested FIND LAST and 1 nested FIND FIRST.



Source Code Reports

Report Category / Name	Matches	Files
CAN-FINDs in a WHERE Clause	0	0
Nested CAN-FIND	0	0
FOR EACH Nested FIND	3	1
FIND By Literal	0	0
Record Delete Directly in a Related Query Loop	3	1
Record Delete Only Directly in a Related Query Loop	3	1
Sequence Usage	0	0
Stored Procedure Usage	0	0
Buffer Copy Breakdown	0	0
Buffer Compare Breakdown	0	0
CONNECT Parameter Types	0	0
CONNECT Parameter Details	0	0

Functions (22 items)
Handles (32 items)
Object Oriented (57 items)
Control Flow/Blocks (89 items)
User Interface (50 items)
Platform Specific (34 items)
Extents (14 items)
Data Types (6 items)
I/O (15 items)
Variable Definitions (29 items)
Shared Resources (17 items)
Sockets (3 items)
XML (3 items)

FOR EACH Nested FIND (3 in 1 files)

Category (2 total)	Matches	%	Files	%
LAST	2	🟩	1	🟩
FIRST	1	🟩	1	🟩

#8 - 11/18/2022 08:01 AM - Greg Shah

Nice! Please post the diff so I can review the approach. It can be tricky to do this kind of static analysis because:

- The nesting can happen at an arbitrarily deep block level but the relationship may still be there.
- The nesting can be inside of called procedures, functions, methods.
- The linking might be done through a passed buffer parameter instead of a more direct linkage.

One other note: I think we also need to match the "FIND unique match" case. This is where the FIND has no FIRST/LAST/NEXT/PREV@/CURRENT qualifier keyword. Instead, it is a match to a unique record based on a literal or the WHERE clause.

Due to the complexities noted above, we would need to do this analysis in some prior step and cannot easily calculate this in a single expression or function. I was thinking that for this purpose it might be easier to implement some runtime tracking that could detect this no matter how the linkage occurs. We discuss this more in #3194. Please read that task and post questions as needed. This #6709 task is meant to implement the tracking/measurement of the #3194 problem so that we can determine if we should work on a solution.

#9 - 11/23/2022 07:40 AM - Dănuț Filimon

- File 6709-rpt.diff added

I took a different approach after taking a look at #3194 and modified how the report is made. I created a new rule set which adds a new annotation at conversion time using walk and ascending rules. In the walk rules, I keep track of FOR EACH occurrences using a stack. In the ascending rules, the annotation is added when a FIND FIRST/LAST/NEXT/PREV/CURRENT/unique match is found and the stack element is removed only when meeting another FOR EACH.

Currently I tested:

- nested find in a deep block level
- multiple nested finds in the same FOR EACH
- finds that are not nested

Greg Shah wrote:

> - The nesting can be inside of called procedures, functions, methods.

The current approach won't work at conversion time, but it should be possible at runtime.

I attached the diff. Please review.

#10 - 11/23/2022 11:16 AM - Greg Shah

Code Review 6709-rpt.diff

Neither annotations/annotations.xml and annotations/annotations_prep.xml can't be used for the nested_find.rules because it is not executed when we run analytics/reporting. You would need to use annotations/early_annotations.xml for anything that needs to be used in reporting.

There is a flaw in the nested_find.rules logic. The only time where there is a nested find is when the FIND is referencing the **same exact buffer** as the containing FOR. Otherwise, these are unrelated queries and should not be linked.

> The current approach won't work at conversion time, but it should be possible at runtime.

Agreed. Since the analytics approach won't be able to detect the wider range of cases, I think the report is not needed. I think it is best to focus on the runtime approach.

#11 - 11/24/2022 05:00 AM - Dănuț Filimon

Greg Shah wrote:

> Neither annotations/annotations.xml and annotations/annotations_prep.xml can't be used for the nested_find.rules because it is not executed when we run analytics/reporting. You would need to use annotations/early_annotations.xml for anything that needs to be used in reporting.

Got it.

> There is a flaw in the nested_find.rules logic. The only time where there is a nested find is when the FIND is referencing the same exact buffer as the containing FOR. Otherwise, these are unrelated queries and should not be linked.

This is my intent, but I wanted to share my work until now. Also, I guess you mean referencing the same buffer in the where clause. FOR EACH tt: FIND FIRST tt. END. doesn't work in 4GL.

> Agreed. Since the analytics approach won't be able to detect the wider range of cases, I think the report is not needed. I think it is best to focus on the runtime approach.

You are right. However, the static analysis can provide **all** find queries inside for each (excluding the ones called inside procedures or dynamic queries). Runtime analysis can cover more cases, but only for the executed scenario. The report can deliver a lower bound which may be higher than the runtime result.

I want to finish the current report ASAP and start working on implementing the runtime tracker.

#12 - 11/24/2022 10:24 AM - Greg Shah

> This is my intent, but I wanted to share my work until now.

Understood. No worries.

> Also, I guess you mean referencing the same buffer in the where clause.

Yes, exactly. This is a kind of implicit join which could be refactored into something more performant.

> I want to finish the current report ASAP and start working on implementing the runtime tracker.

OK

#13 - 11/25/2022 07:11 AM - Dănuț Filimon

- File 6709-rpt-20221125.diff added

I made the changes for the report. Now, it can track FIND statements that reference the same buffer as the FOR EACH in the WHERE clause. I am now going to start implementing the runtime solution.

#14 - 11/25/2022 05:31 PM - Greg Shah

OK, this is close to correct.

1. Please switch out the use of a Stack for the ScopedSymbolDictionary. The idea is similar but the ScopedSymbolDictionary will search up the scopes to find a match. This is important because the match might not be with the nearest enclosing FOR EACH.

2. The following code is matching at the wrong node:

```
      <rule>type == prog.KW_FOR and
            this.descendant(1, prog.KW_EACH) and
            this.nextSibling.type == prog.BLOCK
         <action>recordAst = this.getImmediateChild(prog.RECORD_PHRASE, null)</action>
...
```

The problem here is that it will only match one RECORD_PHRASE while there may actually be many. Also, the check on BLOCK is not needed. The better way to do this is to match on a condition like this:

```
      <rule>relativePath("STATEMENT/KW_FOR/KW_EACH/RECORD_PHRASE")
...
```

Inside that rule (i.e. when that match is true), this and copy will have token type RECORD_PHRASE.

#15 - 12/05/2022 03:51 AM - Dănuț Filimon

- File 6709-rpt-20221205.diff added

Greg Shah wrote:

> 1. Please switch out the use of a Stack for the ScopedSymbolDictionary.

I made the FOR EACH AST the scope and the schemaname of each TEMP-TABLE or BUFFER as symbol.

> 2. The problem here is that it will only match one RECORD_PHRASE while there may actually be many. Also, the check on BLOCK is not needed.

I did the changes to match KW_FOR/RECORD_PHRASE, KW_EACH and RECORD_PHRASE are siblings.

I also made additional changes:

- Included RECORD_PHRASE/BUFFER when adding a symbol to the current scope.
- Swapped the usage of the bufname annotation with schemaname because I could extract the temp-table name from in the case of BUFFER.

Please review the attached diff.

#16 - 12/05/2022 07:34 AM - Greg Shah

Code Review 6709-rpt-20221205.diff

Good, it is getting closer.

1. I don't think we should limit the for each processing to only match temp-tables and buffers. All record phrase targets (tables, temp-tables, buffers, work-tables) are valid.

2. We can't use schemaname because different buffers can share the same schemaname but cannot share the same bufname. If the reference is not to the same buffer then it isn't an implicit join so it isn't a nested case.

3. The the FIND statement WHERE clause processing, you can easily limit the processing to field references using evalLib("fields") or (more explicitly) evalLib("fieldtype", this.type). This is a cleaner approach than searching for a node that has a schemaname that includes a ..

#17 - 12/08/2022 03:24 AM - Dănuț Filimon

- File 6709-rpt-20221208.diff added

Greg Shah wrote:

> 1. I don't think we should limit the for each processing to only match temp-tables and buffers. All record phrase targets (tables, temp-tables, buffers, work-tables) are valid.

I made the changes to include tables and work-tables.

> 2. We can't use schemaname because different buffers can share the same schemaname but cannot share the same bufname. If the reference is not to the same buffer then it isn't an implicit join so it isn't a nested case.
> 3. The the FIND statement WHERE clause processing, you can easily limit the processing to field references using evalLib("fields") or (more explicitly) evalLib("fieldtype", this.type). This is a cleaner approach than searching for a node that has a schemaname that includes a ..

Since schemaname should not be used, I returned to bufname and instead of checking if the annotation is present, I used evalLib("fields") to make sure that a field is evaluated.

I attached the diff with the mentioned changes.

#18 - 12/08/2022 08:45 AM - Dănuț Filimon

```
         AdaptiveQuery query4 = new AdaptiveQuery();

         forEach("loopLabel13", new Block((Init) () -> 
         {
            query4.initialize(wtcustomer, ((String) null), null, "wtcustomer.recid asc");
         },
         (Body) () ->
         {
            query4.next();
            new FindQuery(tt1, "tt1.fieldTt11 < ?", null, "tt1.fieldTt11 asc", new Object[]
            {
               (P2JQuery.Parameter) () -> wtcustomer.getCustnum() // can't retrieve buffer from lambda
            }).next();
         }));
```

This is how a FindQuery is generated as a nested find. The problem is that I can't retrieve the buffer from the **where** clause. A solution for this would be to change the lambda to FieldReference.

#19 - 12/08/2022 04:32 PM - Greg Shah

I think FieldReference is less clean looking and it will be slower. In #4928, we do have some ideas to improve it using lambdas but it won't ever be as clean as the current approach. We could implement this as an optional conversion which could then be used by the runtime for detection of nesting. If that was only done for this kind of testing, then it might be OK.

Eric: What other ramifications are there which we need to consider?

#20 - 12/08/2022 04:40 PM - Greg Shah

> 1. I don't think we should limit the for each processing to only match temp-tables and buffers. All record phrase targets (tables, temp-tables, buffers, work-tables) are valid.

I made the changes to include tables and work-tables.

Instead of this:

```
         <action>recordAst = null</action>
         <rule>this.getImmediateChild(prog.TEMP_TABLE, null) != null
            <action>recordAst = this.getImmediateChild(prog.TEMP_TABLE, null)</action>
         </rule>
         <rule>this.getImmediateChild(prog.BUFFER, null) != null
            <action>recordAst = this.getImmediateChild(prog.BUFFER, null)</action>
         </rule>
         <rule>this.getImmediateChild(prog.TABLE, null) != null
            <action>recordAst = this.getImmediateChild(prog.TABLE, null)</action>
         </rule>
         <rule>this.getImmediateChild(prog.WORK_TABLE, null) != null
            <action>recordAst = this.getImmediateChild(prog.WORK_TABLE, null)</action>
         </rule>
```

You can just do this:

```
         <action>recordAst = this.getChildAt(0)</action>
```

There are only those 4 possible cases and they are always the first child of the RECORD_PHRASE.

Otherwise the proposed changes look good. If they pass testing, then you can commit them in 3821c.

#21 - 12/12/2022 04:29 AM - Dănuț Filimon

Greg Shah wrote:

> You can just do this:
>
> ```
>     <action>recordAst = this.getChildAt(0)</action>
> ```

Did so.

> If they pass testing, then you can commit them in 3821c.

I created a report before the modification above was introduced. The report was generated successfully, but no nested finds were found in the project. I plan to introduce a few non-related nested finds in the project and generate another report soon.

Committed 3821c/rev.14436. Added FOR EACH nested FIND report.

#22 - 12/12/2022 08:17 AM - Greg Shah

I've uncommitted 3821c rev 14436. It causes the following regression in conversion for a large customer application. See #6851-153.

Danut, please address the bug and make sure to test conversion with that application.

#23 - 12/14/2022 10:01 AM - Dănuț Filimon

Looks like the problem was caused by a FOR FIRST table statement. Fixed by checking if the previous sibling of the RECORD_PHRASE is KW_EACH when getting the recordAst and saving a symbol. I took the individual statement from the file mentioned in #6851-153 and converted a similar example separately with no problems. I am currently testing conversion of the specified application and will make an update when it's done.

#24 - 12/16/2022 03:31 AM - Dănuț Filimon

Greg Shah wrote:

> Danut, please address the bug and make sure to test conversion with that application.

The bug is fixed and the conversion was successful. Should I commit the changes?

I am currently trying to convert

```
(P2JQuery.Parameter) () -> wtcustomer.getCustnum()
```

into

```
new FieldReference(wtcustomer, "custnum")
```

I managed to find where the cast to P2JQ.Parameter is made, but it wasn't the solution that I needed. I was left with wtcustomer.getCustnum(). My question is where in the rules is the **CustNum getter** added? I want to keep wtcustomer and custnum separatelly so i can create a FieldReference.

#25 - 12/16/2022 08:30 AM - Greg Shah

> The bug is fixed and the conversion was successful. Should I commit the changes?

Please post an updated patch for review.

> My question is where in the rules is the CustNum getter added? I want to keep wtcustomer and custnum separatelly so i can create a FieldReference.

We emit the getter or the FieldReference in convert/database_references.rules. I think the most likely scenario here is to use the accessor flag to force FieldReference to be used. Make sure to do this based on a flag that by default is off. In other words, we only want to turn this on when an explicit flag is set.

#26 - 12/16/2022 09:53 AM - Dănuț Filimon

- File 6709-20221216.patch added

Greg Shah wrote:

> Please post an updated patch for review.
>
> We emit the getter or the FieldReference in convert/database_references.rules. I think the most likely scenario here is to use the accessor flag to force FieldReference to be used. Make sure to do this based on a flag that by default is off. In other words, we only want to turn this on when an explicit flag is set.

This is what I was looking for. Thank you.

#27 - 12/16/2022 10:59 AM - Greg Shah

I'm OK with the 6709-20221216.patch.

You've tested conversion for both the large customer application and also Hotel GUI?

#28 - 12/19/2022 03:41 AM - Dănuț Filimon

Greg Shah wrote:

> You've tested conversion for both the large customer application and also Hotel GUI?

Yes, both applications converted successfully.

Can I commit the changes to 3821c?

#29 - 12/19/2022 06:09 PM - Greg Shah

Yes.

#30 - 12/20/2022 05:42 AM - Dănuț Filimon

Committed 3821c/rev.14460. Added nested find annotation and report. Any annotated FIND that is nested in a FOR EACH will be displayed under FOR EACH Nested FIND when generating a report and classified based on the keyword used.

#31 - 12/22/2022 07:02 AM - Dănuț Filimon

- File 6709-20221222.diff added

I managed to convert Hotel GUI and a few smaller tests and use FieldReference. I actually didn't make use of convert/database_references.rules because the accessor annotation would make the query components also use FieldReference and additional changes proved really difficult.

I found out that the related_buffer annotation seems to influence the usage of FieldReference. As suggested in #6709-25, I added a flag called enable_find_field_reference in the directory.hints file and made changes in annotations/index_selection.rules to use the hint to set the related_buffer annotation to **true** in all FindQueries.

I've tested conversion on a customer application overnight at least 2 times and fixed small mistakes. I also managed to test runtime tracking of nested finds. My approach was to use the offEndQueries value from the TransactionManager, but the set is actually cleared after the block.init() is executed, while I need it in the body which contains the FindQuery. I created a list in BlockDefinition to store the values of offEndQueries in the top block of the WorkArea and make use of them when checking if the record buffer of the FindQuery matches any of the record buffers of the components. The list is then cleared when the scope is closed.

When I wrote the implementation for the runtime tracking of nested finds, I looked over 3194a/rev.11127 and borrowed a similar idea, but it's a bit more general since the idea was to only track nested finds. I attached a diff with the latest changes, please take a look at it and tell me if any improvements can be made.

#32 - 12/22/2022 08:40 AM - Alexandru Lungu

I will try to patch a customer application I have on my development environment to test the changes. If everything is right, I will do some preliminary tracking on that application.

#33 - 12/22/2022 09:49 AM - Greg Shah

Code Review 6709-20221222.diff

The conversion changes seem reasonable.

Do we really need to implement this as part of the query-off-end processing? All of this behavior is deeply specific to the persistence layer. It seems like the BufferManager or something else in persistence should easily be able to provide a list of the active FOR EACH queries. Extending the TransactionManager with even more query-specific logic is not good. At a minimum, we would only want this runtime processing when a flag is set in the directory. If it is not active, then there should be no processing and no impact of the tracking.

Although we do have some off-end processing inside the generic transaction/block processing, **I really want to remove it**. If this "get the list of all for each queries" processing really needs to be tracked during the block processing, I would hope we already expose the block/transaction notifications that are needed.

The matching logic in the FindQuery seems expensive. It seems like the inverse of what we need. We already have access to the buffer being referenced. Can't we just ask the buffer if it is associated with a FOR EACH query? Looping through all FOR EACH queries to see if this buffer is in the list is a lot of unnecessary work that some simple state in the buffer could answer. If we need more state in the buffer to store this, that seems a better way.

#34 - 12/22/2022 09:50 AM - Greg Shah

- Related to Support #7004: eliminate the off-end processing in the TransactionManager and minimize the processing of off-end added

#35 - 12/22/2022 09:51 AM - Greg Shah

Eric: Please review and advise.

#36 - 01/06/2023 02:51 AM - Dănuț Filimon

- File 6709-20230106.patch added

I've changed the way tracking is done based on #6709-33. The only change I've done to TransactionManager is to retrieve offEndQueries and save the value in a Map from BufferManager until the **for each body** closes. I also included a flag in directory.xml, called enable_find_field_reference, which by default is false and is used to toggle the runtime tracking of nested finds. I also created a MBean that can be used to track the counter of FindQueries that were initialized. I tested Hotel GUI and a customer application (~30 minutes) and the MBean didn't increase, I made sure it worked using a small test.

Please review the attached patch.

Files

report-nested-find.png	581 KB	11/18/2022	Dănuț Filimon
6709-rpt.diff	9.08 KB	11/23/2022	Dănuț Filimon
6709-rpt-20221125.diff	9.64 KB	11/25/2022	Dănuț Filimon
6709-rpt-20221205.diff	10.9 KB	12/05/2022	Dănuț Filimon
6709-rpt-20221208.diff	11.1 KB	12/08/2022	Dănuț Filimon
6709-20221216.patch	10.5 KB	12/16/2022	Dănuț Filimon
6709-20221222.diff	11 KB	12/22/2022	Dănuț Filimon
6709-20230106.patch	18.7 KB	01/06/2023	Dănuț Filimon