Base Language - Bug #2069

Accumulator temp table field disambiguation

02/28/2013 05:58 AM - Costin Savin

Status:	Closed	**Start date:**	02/28/2013
Priority:	Normal	**Due date:**	
Assignee:	Constantin Asofiei	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	Conversion Support for Server Features		
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

If we create a temp table with a field name that coincides with a field from an existent permanent table and use accumulate out of the table context conversion won't be able to handle it.

History

#1 - 02/28/2013 06:09 AM - Costin Savin

For the following testcase we define a temp table fit a field name that already exists in a permanent table and use it out of context in an accumulate like:

```
def temp-table tt1 field book-title as char.
find first book.
for each tt1 by tt1.book-title:
   accumulate tt1.book-title (count).
end.

message accum count book-title.
```

The following error will occur:

```
EXPRESSION EXECUTION ERROR:
---------------------------
throwException("Unmatched expression in ACCUM function", this)
^  { Unmatched expression in ACCUM function [KW_COUNT id <236223201346> 7:15] }
---------------------------
ERROR:
java.lang.RuntimeException: ERROR!  Active Rule:
------------------------
      RULE REPORT
------------------------
Rule Type :   WALK
Source AST:  [ count ] BLOCK/STATEMENT/KW_MSG/CONTENT_ARRAY/EXPRESSION/FUNC_POLY/AGGREGATE/KW_COUNT/ @7:15 {23
6223201346}
Copy AST  :  [ count ] BLOCK/STATEMENT/KW_MSG/CONTENT_ARRAY/EXPRESSION/FUNC_POLY/AGGREGATE/KW_COUNT/ @7:15 {23
6223201346}
Condition :  throwException("Unmatched expression in ACCUM function", this)
Loop      :  false
--- END RULE REPORT ---
```

#2 - 02/28/2013 11:02 AM - Constantin Asofiei

- Assignee changed from Costin Savin to Constantin Asofiei

After the note 75 in the 1985, looks like this is a true buffer disambiguation problem, which can be duplicated using:

```
def temp-table tt1 field book-title as char.

for each book:
   message book.book-title.
end.

for each tt1 by tt1.book-title:
   accumulate tt1.book-title (count).
end.

find first tt1.
message accum count book-title.
```

The idea was expand the scope for tt1.

Now, I think the clue for this is point 5 in the note:

> 5. Should the references inside an ACCUM be a "no reference" instead of a "free reference"? After all, we are just trying to match up the "name" of the original accumulator, and we are not really referencing the field. Or are we?

For now, I will try to avoid the red pill <g>, as maybe there is a simpler explication: like UNDERLINE field, maybe accum function should produce no-reference's instead of free-references.

#3 - 03/03/2013 06:40 AM - Constantin Asofiei

First of all, looks like ACCUM function should not produce free-references, because if it did produce, then the code at note 2 should have worked in 4GL without the FIND statement (which produced a free reference and expanded the scope). Also, P2J doesn't properly compute the scope for tt1, in the case where it should be at the FOR EACH tt1 block:

```
{} Line Blk
-- ---- ---
      1     def temp-table tt1 field book-title as char.
      2
      3   1 for each book:
```

```
      4     end.
      5
      6   1 for each tt1 by tt1.book-title:
      7   1   accumulate tt1.book-title (count).
      8     end.
      9
 sc4.p                                03/03/2013 11:24:58   PROGRESS(R) Page 2

     File Name       Line Blk. Type   Tran            Blk. Label
-------------------- ---- ---------- ---- --------------------------------
sc4.p                   0 Procedure   No
sc4.p                   3 For         No
    Buffers: p2j_test.Book

sc4.p                   6 For         No
    Buffers: tt1
```

In converted code, the RecordBuffer.openScope call for tt1 is at the external procedure.

Second, I'm pulling a rabbit out of the rabbit hole, please let me know if you think it looks like the right one. Assuming we have the code at note 2, there is no way to reference the unqualified 'book-title' field without using the accum function. If we replace the last statement with:

```
message book-title.
```

4GL will show an ambiguity compile error.
I've tested how the accumulator is referenced, and my conclusion is it uses the disambiguated field name. Also, if we switch the "book" and "tt1" tables, the behavior is the same (so IMO it's a matter of name collisions in buffers).

Where I think the problem is. When the find first tt1 statement is executed in the code at note 2, the 4GL promotes the scope of tt1 to the root procedure:

```
{} Line Blk
-- ---- ---
      1     def temp-table tt1 field book-title as char.
      2
      3   1 for each book:
      4   1    message book.book-title.
      5     end.
      6
      7   1 for each tt1 by tt1.book-title:
      8   1    accumulate tt1.book-title (count).
      9     end.
     10
     11     find first tt1.
     12     message accumulate count book-title.
 sc4.p                                03/03/2013 09:05:54   PROGRESS(R) Page 2

     File Name       Line Blk. Type   Tran            Blk. Label
-------------------- ---- ---------- ---- --------------------------------
sc4.p                   0 Procedure   No
    Buffers: tt1

sc4.p                   3 For         No
    Buffers: p2j_test.Book

sc4.p                   7 For         No
```

without the FIND statement, the scope of the tt1 buffer would be at its FOR EACH block:

```
{} Line Blk
-- ---- ---
      1     def temp-table tt1 field book-title as char.
      2
      3   1 for each book:
      4   1    message book.book-title.
      5     end.
      6
      7   1 for each tt1 by tt1.book-title:
      8   1    accumulate tt1.book-title (count).
```

```
      9     end.
     10
     11     /*find first tt1.
     12     message accumulate count book-title. 4GL reports ambiguity for this statement*/
 sc4.p                                03/03/2013 09:07:11   PROGRESS(R) Page 2

     File Name       Line Blk. Type   Tran            Blk. Label
-------------------- ---- ---------- ---- --------------------------------
sc4.p                   0 Procedure   No
sc4.p                   3 For         No
    Buffers: p2j_test.Book

sc4.p                   7 For         No
    Buffers: tt1
```

Now, some notes about how the structure of this program affects scoping. In SchemaDictionary, there are these scope types:

1. the schema global scope, in position 0; this includes all permanent tables
2. the user global scope, with the temp-table definitions (and buffers too I think).
3. the external program scope

- The FOR EACH book statement will add in this scope all book's fields, after it has finished (see the inner_block's sym.deleteSchemaScope(true) call, which propagates the fields added in the current scope to the caller's scope.
- when FOR EACH tt1 statement finishes, inner_block's sym.deleteSchemaScope(true) will be called, but no propagation will be done in the external program scope for the tt1.book-title field, as there is already the book.book-title field in the external program scope. I can't tell if this was intended or not.
- when the FIND FIRST tt1 statement is executed, this calls record_phrase[false, false]. I don't like this part, as 4GL suggests that the statement does propagate the tt1 buffer to the external procedure. What I've tried was this in find_stmt rule:

```
      {
         sym.addSchemaScope(false); /* open a new scope for the find's target */
      }
      record_phrase[true, false] /* force the target buffer to be promoted to current scope */
      DOT!
      {
         sym.deleteSchemaScope(true); /* delete the scope and let the buffer name propagate */
      }
```

and I've hit the same problem as for the FOR EACH tt1 statement - the tt1.book-title does not propagate because the external program's scope already has a field named book-title.
A dirty fix for this was to modify SchemaDictionary.propagate, to replace the old value in the namespace with the new value, if already exists. But this doesn't solve us cases where we have FIND statements for both the book and tt1 buffers, at the end of the program:

```
find first book.
find first tt1.
message accum count book-title. /* 4GL generated ambiguity error at compile */
```

Using my SchemaDictionary.propagate fix, P2J will disambiguate book-title in the accum statement to tt1.book-title, when it should have failed. More, for the FOR EACH tt1 case, it will add the tt1.book-title to the caller's scope, when 4GL doesn't seem to do so (thus P2J will convert the program even without the FIND FIRST statement in note 2).
But, if we force the ACCUM function to not produce free-references, then the accumulator expression will still not be found, because currently the match is done using the (refid, schemaname) pair.

#4 - 03/03/2013 12:32 PM - Greg Shah

Good information in note 3.

1. You have proven that the ACCUM function parameters are not free references, they are a "no_reference". This can be implemented in the field portion of the buffer_scoping_reference_type rule in common-progress.rules. Please go ahead with that change.

> But, if we force the ACCUM function to not produce free-references, then the accumulator expression will still not be found, because currently the match is done using the (refid, schemaname) pair.

2. I have isolated an example that duplicates the structure of the original enough to allow the unqualified field reference to work in the 4GL:

```
def temp-table real      field f1 as char.
def temp-table conflict  field f1 as char.
def temp-table container field num as int.

for each container:
   for each conflict:
   end.

   create real.
   real.f1 = "something".
end.

for each real by real.f1:
   accumulate real.f1 (count).
end.

message accum count f1.
```

Here is the listing:

```
...nsion_conflicts.p                  03/03/2013 17:21:37   PROGRESS(R) Page 1

{} Line Blk
-- ---- ---
      1     def temp-table real      field f1 as char.
      2     def temp-table conflict  field f1 as char.
      3     def temp-table container field num as int.
      4
      5   1 for each container:
      6   2    for each conflict:
      7   1    end.
      8   1
      9   1    create real.
     10   1    real.f1 = "something".
     11     end.
     12
     13   1 for each real by real.f1:
     14   1    accumulate real.f1 (count).
     15     end.
     16     message accum count f1.
 ...nsion_conflicts.p                   03/03/2013 17:21:37   PROGRESS(R) Page 2

     File Name       Line Blk. Type   Tran            Blk. Label
-------------------- ---- ---------- ---- --------------------------------
...nsion_conflicts.p    0 Procedure  No
    Buffers: real

...nsion_conflicts.p    5 For        No
    Buffers: container

...nsion_conflicts.p    6 For        No
    Buffers: conflict

...nsion_conflicts.p   13 For        No
```

It looks to me like there is some limit to the propagation behavior. Perhaps the presence of the "create real" following the for each conflict causes the previously propagated conflict values to be cleared and the real table to be added back. Note that there is no FIND here.

Please review the implicit scope expansion rules to see if you can explain this behavior with what we currently know. If not, then we are probably dealing with a new case, which will have to be the subject of some experiments to determine the new rule(s). Based on this, I hope we can find a safe change to the propagation behavior to allow this to parse. We would also need to change our buffer scoping rule-sets to properly calculate this result.

#5 - 03/03/2013 03:40 PM - Constantin Asofiei

Some final notes for today:

1. the CREATE real statement produces a free reference, same as a FIND FIRST real. If we replace CREATE with FIND, then a similar listing is produced by 4GL:

```
{} Line Blk
-- ---- ---
      1     def temp-table real      field f1 as char.
      2     def temp-table conflict  field f1 as char.
      3     def temp-table container field num as int.
      4   1 for each container:
      5   2    for each conflict:
      6   1    end.
      7   1
      8   1    find first real.
      9     end.
     10
     11   1 for each real by real.f1:
     12   1    accumulate real.f1 (count).
     13     end.
     14
     15     message accum count f1.
 sc5.p                                03/03/2013 18:00:14   PROGRESS(R) Page 2

     File Name       Line Blk. Type   Tran            Blk. Label
-------------------- ---- ---------- ---- --------------------------------
sc5.p                   0 Procedure  No
    Buffers: real

sc5.p                   4 For        No
    Buffers: container

sc5.p                   5 For        No
    Buffers: conflict

sc5.p                  11 For        No
```

2. when parsing your example, P2J behaves similar with my previos example:

- FOR EACH conflict propagates conflict to the FOR EACH container's scope
- CREATE real tries to add real.f1 to the FOR EACH conflict's scope (via a record[true, false, false] call by the create_stmt rule), but as real is already marked as promoted (the mark was done by the DEFINE TEMP-TABLE statement), it does not add it.
- when the FOR EACH container ends, it propagates conflict.f1 to the external program's scope.
- when the FOR EACH real block ends, parser again tries to propagate f1 field for real.f1 to the external procedure, but as f1 for conflict.f1 is already there, it does not do so.

Another interesting fact is that if we replace temp-tables with permanent tables, as in this example:

```
{} Line Blk
-- ---- ---
      1   1 for each book:
      2   2    for each pers-addr:
      3   1    end.
      4   1
      5   1    create person.
      6   1    person.ssn = "".
```

```
      7     end.
      8
      9   1 for each person by person.emp-num:
     10   1    accumulate person.emp-num (count).
     11     end.
     12
     13
     13     message accum count emp-num.
 sc6.p                                03/03/2013 19:48:43   PROGRESS(R) Page 2

     File Name       Line Blk. Type   Tran            Blk. Label
-------------------- ---- ----------- ---- --------------------------------
sc6.p                   0 Procedure   No
    Buffers: p2j_test.Person

sc6.p                   1 For         Yes
    Buffers: p2j_test.Book

sc6.p                   2 For         No
    Buffers: p2j_test.Pers-Addr

sc6.p                   9 For         No
```

P2J behaves different. The resulted RecordBuffer.openScope for the perm tables are in the correct places (as in the listing), while for the temp-table example, all the temp-buffers are scoped to the external procedure in the resulted code. Considering that IMO the temp-buffers are not processed correctly by the parser, I'm switching to the perm table example, until I understand why and how the free reference should get promoted. In my perm-table example, person is the "real" buffer, while pers-addr is the "conflict" buffer. Now, some more about where I think P2J is wrong. In progress.g, for the inner_block rule, there is this javadoc comment:

```
 Note that at this time, there is only a simple imitation of the implicit expansion of a record scope from one
 block to another. This is done by the record rule by ALWAYS promoting a table to the current scope. A more co
mplicated solution may be necessary but it is not yet known how much this is actually in use.
```

I think this contradicts with this note, in the Record Scopes/Weak Reference documentation:

```
If a weak reference is not nested inside a strong scope, it will create a new buffer and scope that buffer to
the block definition which made the weak reference, BUT this will only happen if no unbound free references to
 the same buffer exist (see Free Reference below for details).
```

The note in the documentation suggests that conflict buffer (pers-addr buffer in perm-table example) with the weak reference must not propagate to the caller.
More, after I experimented a little using examples with or without the free-reference added for the real (or person buffer), looks like the scope is expanded to the external procedure (or I think better said to the nearest block), to enclose both the free and the weak reference. My experiments included:

1. two weak-references, enclosed in different blocks:

```
{} Line Blk
-- ---- ---
      1   1 for each book:
      2   2    for each pers-addr:
      3   1    end.
      4   1
      5   2    for each person:
      6   1    end.
      7   1
      8     end.
      9
     10   1 for each vehicle:
     11   2   for each person by person.emp-num:
     12   1   end.
     13     end.
 sc6.p                                03/03/2013 20:19:47   PROGRESS(R) Page 2

     File Name       Line Blk. Type   Tran            Blk. Label
-------------------- ---- ----------- ---- --------------------------------
sc6.p                   0 Procedure   No
sc6.p                   1 For         No
    Buffers: p2j_test.Book
```

```
sc6.p                    2 For          No
    Buffers: p2j_test.Pers-Addr

sc6.p                    5 For          No
    Buffers: p2j_test.Person

sc6.p                   10 For          No
    Buffers: p2j_test.vehicle

sc6.p                   11 For          No
    Buffers: p2j_test.Person
```

2. if we add a free reference, the person buffer will get promoted to a block to enclose them all:

```
-- ---- ---
      1   1 repeat:
      2   1
      3   2 for each book:
      4   3    for each pers-addr:
      5   2    end.
      6   2
      7   2    create person.
      8   1 end.
      9   1
     10   2 for each vehicle:
     11   3   for each person by person.emp-num:
     12   2   end.
     13   1 end.
     14   1
     15     end.
 sc6.p                                  03/03/2013 20:21:04   PROGRESS(R) Page 2

     File Name       Line Blk. Type   Tran            Blk. Label
-------------------- ---- ---------- ---- --------------------------------
sc6.p                   0 Procedure   No
sc6.p                   1 Repeat      No
    Buffers: p2j_test.Person

sc6.p                   3 For         Yes
    Buffers: p2j_test.Book

sc6.p                   4 For         No
    Buffers: p2j_test.Pers-Addr

sc6.p                  10 For         No
    Buffers: p2j_test.vehicle

sc6.p                  11 For         No
```

3. if there are no weak references for person, then the free reference will not propagate:

```
sc6.p                                  03/03/2013 20:22:05   PROGRESS(R) Page 1

{} Line Blk
-- ---- ---
      1   1 repeat:
      2   1
      3   2 for each book:
      4   3    for each pers-addr:
      5   2    end.
      6   2    create person.
      7   1 end.
      8   1
      9     end.
 sc6.p                                  03/03/2013 20:22:05   PROGRESS(R) Page 2

     File Name       Line Blk. Type   Tran            Blk. Label
-------------------- ---- ---------- ---- --------------------------------
sc6.p                   0 Procedure   No
sc6.p                   1 Repeat      No
```

```
sc6.p                    3 For          Yes
    Buffers: p2j_test.Person
             p2j_test.Book

sc6.p                    4 For          No
    Buffers: p2j_test.Pers-Addr
```

In terms of scope compuation, P2J behaves nicely in all cases. But the problem is this is done by the TRPL rules, and not by the parser. The most important notes I think are, when speaking of field name disambiguation:

1. inner_block rule must not propagate the weak references in all cases, it must do so only if a free reference was previously encountered
2. same for free references, they must propagate only if there are weak references too
3. SchemaDictionary needs support to track free/weak references.

#6 - 03/04/2013 09:41 AM - Greg Shah

Please prepare the ACCUM no_reference and CAN-FIND changes and upload the update ASAP. Both of these seem to be correct and in fact, the CAN-FIND change is much needed right now as I suspect it solves all but one of the buffer scoping problems. You can go ahead and put that update through conversion testing and I will check on the recent buffer scope problems in #1985.

#7 - 03/04/2013 09:57 AM - Constantin Asofiei

About the ACCUM no_reference: after I let this be a no_reference, then the "refid" annotation will not be emitted for it. Thus, as annotations/accumulate.rules identifies a field using the (refid, schemaname) pair, it will need some changes. I'm checking too, but do you see any reason why matching the fields using the schemaname only to be incorrect?

#8 - 03/04/2013 10:06 AM - Constantin Asofiei

- File ca_upd20130304a.zip added

This update:

- makes the field in accum function to be a no-reference.
- fixes CAN-FIND function, when its WHERE clause has unqualified fields (it matches them first with the target buffer).

#9 - 03/04/2013 11:15 AM - Greg Shah

The changes look good, though we may have to work off something different than only schemaname.

```
def temp-table tt field num as int.

def buffer b1 for tt.
def buffer b2 for tt.

for each b1:
   accumulate b1.num (total).
end.

for each b2:
```

```
   accumulate b2.num (total).
end.

message accum total num.
```

This generates this error:

```
** Unable to find ACCUM expr in an earlier DISPLAY or ACCUM
statement--try qualifying field names with table name. (364)
**  Could not understand line 14. (196)
```

This works:

```
def temp-table tt field num as int.

def buffer b1 for tt.
def buffer b2 for tt.

for each b1:
   accumulate b1.num (total).
end.

for each b2:
   accumulate b2.num (total).
end.

message accum total b1.num.
message accum total b2.num.
```

Perhaps we need to match differently.

#10 - 03/04/2013 11:41 AM - Constantin Asofiei

If the parser gets the schemaname right, in case of unqualified field and compilable 4GL code, I don't think we will have a problem, as the schemaname annotation will contain the fully qualified field name.

And, my intuition tells me that our problem exposed by the accum function (which produces no-references) is for any statement/function which produces no-references:

```
def temp-table real      field f1 as char label "bar".
def temp-table conflict  field f1 as char label "foo".
def temp-table container field num as int.

for each container:
   for each conflict:
   end.

   create real.
   real.f1 = "something".
end.

for each real by real.f1:
end.

form f1 with frame frame1. /* this is a no-reference */
view frame frame1. /* in 4GL, this shows "bar" label; P2J shows the "foo" label, thus it uses the "conflict.f1
" field, instead of "real.f1" */
```

I'm more and more inclined to conclude that our buffer scope resolution is done too late, in the TRPL rules, when it should have been done in the parser. After debugging the parser in terms of buffer scope promotion and especially propagation, the way it acts now seems very flawed to me.

I will try (as soon as I can) to come with an approach of tracking the weak/free/strong reference scopes, and merge them using the rules described in our documentation, during parser phase; but, what I don't like is that I think we will end up with the algorithm for buffer scope resolution being duplicated in the parser phase.

#11 - 03/04/2013 12:09 PM - Constantin Asofiei

The 0304a.zip passed conversion regression testing, but there are lots of buffer scope-related changes, as the accum function now produces only no-references. Runtime regression testing should be done with only this update, so that all possible problems can be linked to it.

#12 - 03/04/2013 12:18 PM - Greg Shah

I don't think schemaname alone will work.

In my example, both fields have a schemaname of "tt.num" but the bufname will differ ("b1" vs "b2"). Actually, the name annotation is "b1.num" or "b2.num" but I don't think this can be relied upon (e.g. I think this corresponds to the text name which can be different for different references to the same field + buffer combination). A combination of schemaname and bufname should do the job.

Unless I am missing something, we should go ahead and fix this now.

> And, my intuition tells me that our problem exposed by the accum function (which produces no-references) is for any statement/function which produces no-references:

Yes, this is correct.

> I'm more and more inclined to conclude that our buffer scope resolution is done too late, in the TRPL rules, when it should have been done in the parser. After debugging the parser in terms of buffer scope promotion and especially propagation, the way it acts now seems very flawed to me.

Also true. But I fear changing things at this point. It will mean a great deal of re-implementation of the buffer scoping processing, which will be risky (prone to errors).

We have always worried that the Progress compiler handles buffer scoping as it goes (in the main pass) and that our use of scopes, promotion and the tiered namespace processing is just a patchwork that doesn't fully match the buffer scoping behavior of the 4GL.

> what I don't like is that I think we will end up with the algorithm for buffer scope resolution being duplicated in the parser phase.

If we do push buffer scoping into parse time, then we will remove it from annotations.

The key question: can we find a less risky patch that will reliably solve this problem for now? I know, "for now" always seems to end up as "forever". But I hesitate to de-rail milestone 4.

#13 - 03/04/2013 12:25 PM - Constantin Asofiei

> In my example, both fields have a schemaname of "tt.num" but the bufname will differ ("b1" vs "b2"). Actually, the name annotation is "b1.num" or "b2.num" but I don't think this can be relied upon (e.g. I think this corresponds to the text name which can be different for different references to the same field + buffer combination). A combination of schemaname and bufname should do the job.
>
> Unless I am missing something, we should go ahead and fix this now.

OK, I see your point now, I will try and fix this tonight.

> The key question: can we find a less risky patch that will reliably solve this problem for now? I know, "for now" always seems to end up as "forever". But I hesitate to de-rail milestone 4.

My approach would be this: leave the actual buffer scope resolution in the TRPL rules (I mean, the BUFFER_SCOPE node creation), and patch the parser/SchemaDictionary to handle the weak+free reference cases which troubles us, as currently all propagation for weak scopes is incorrect, in the parser phase. This will solve us (I hope) the field name disambiguation, but will leave all the buffer scope resolution work to the TRPL rules.

#14 - 03/04/2013 01:27 PM - Constantin Asofiei

- File ca_upd20130304b.zip added

Looks like the matching in accumulators needs to be done using the bufname too, attached update does this. I'm putting it into conversion regression testing now.

#15 - 03/04/2013 01:52 PM - Greg Shah

The code looks good.

It also fixes all remaining issues in #1985 notes 93, 104-109 and 111. These programs all convert successfully now.

#16 - 03/04/2013 02:48 PM - Constantin Asofiei

Conversion regression testing has finished, the note about runtime regression testing still stands, with the addition that some programs (which have changed related to accumulators too, not just buffer scoping) I think will be best to be tested by hand, if they are not reached by the regression harness.

#17 - 03/04/2013 03:01 PM - Greg Shah

Check it it and distribute it. I will make sure runtime regression testing is run separately. What programs should be tested by hand?

#18 - 03/04/2013 03:17 PM - Constantin Asofiei

Committed to bzr revision 10238.

These files have changes which are not limitted to just buffer scope:

```
co/Sa14R.java
est/Est02R.java
item/ItemCC.java
job/Job26R.java
job/Job72cR.java
job/Job72R.java
so/Serv03R.java
```

#19 - 03/04/2013 03:39 PM - Constantin Asofiei

Some news about the field name disambiguation issue. In SchemaDictionary.promoteEntry, there is this test:

```
3630: // Buffers associated with a weak scope are not propagated.
      if (noProp && fromNode.getType() == BUFFER)
      {
         scopes.get(scopes.size() - 1).setWeakReference(fromNode);

         noPropagate = true;
      }
```

This code tests only for BUFFER nodes, and excludes tables, temp-tables, work-tables. Was this by intent? Because, if I change it to:

```
3630: // records associated with a weak scope are not propagated.
      int fromType = fromNode.getType();
      if (noProp && fromType >= BEGIN_RECORDTYPES && fromType <= END_RECORDTYPES)
      {
         scopes.get(scopes.size() - 1).setWeakReference(fromNode);

         noPropagate = true;
      }
```

My test which uses perm tables passes conversion and disambiguates the no-reference correctly. I still have problems with the temp-table tests (the DEFINE TEMP-TABLE doesn't allow the real buffer to be promoted), but please let me know if the above approach looks correct.

#20 - 03/04/2013 03:59 PM - Eric Faulhaber

Constantin Asofiei wrote:

> This code tests only for BUFFER nodes, and excludes tables, temp-tables, work-tables. Was this by intent?

I doubt the exclusion was intentional. I suspect we were trying to solve a particular problem with buffers at the time this was written. I guess your fix is OK. Run it through conversion regression testing to be sure, of course.

#21 - 03/04/2013 04:38 PM - Constantin Asofiei

- File ca_upd20130304c.zip added

This fix attempt does the following:

1. fixes #2059 's ecf_upd20130303b.zip, to include temp-tables too
2. fixes a bug in promoteEntry, to include all record types when checking for a weak scope
3. IMO, the SchemaDictionary.promoted set should be actually saved per each scope, and not just for each top level scope (this was why the real buffer's free reference was not being promoted to the current scope and after that propagated up the stack, as the SchemaDictionary.promoted set already contained the real buffer, from the define temp-table statement).

I'm putting this through conversion regression testing. It's a weird place down here in the rabbit hole...

#22 - 03/04/2013 07:01 PM - Greg Shah

Does this resolve #1985 note 64 or is this just more fixes on the way to the Mad Hatter's tea party?

#23 - 03/04/2013 10:48 PM - Constantin Asofiei

With these changes, pmcli/ converts fully. Also, MAJIC converts fully, but there is a CAN-FIND change in a single file which I'm not sure yet why is happening, and it looks like a regression. Anyway, I feel I'm on the right path.

#24 - 03/04/2013 11:41 PM - Constantin Asofiei

- File ca_upd20130305a.zip added

This makes the INPUT function to produce no-references. About 04c.zip: my test was without 04b.zip update, to better isolate any changes, but it looks like 04b.zip's CAN-FIND change is needed by 04c.zip. Anyway, my plan is this:

1. convert with 04c.zip + 04b.zip, and compare the results with the 04b.zip's generated sources - there should be no differences
2. convert 04c.zip with 04b.zip's part related only to CAN-FIND, and compare the results with the generated sources prior to 04b.zip - there should be no differences

#25 - 03/05/2013 05:14 AM - Constantin Asofiei

In the end, I ended up testing (using 10237 as baseline, but this should make no difference):

1. 05a (INPUT function no_reference) - some changes related to index changes, see separate mail (I think the changes are OK)
2. 04c alone - it produced a regression in one file, related to CAN-FIND
3. 04b_no_accum alone - it has a regression in one file (wrong buffer is used as far as I can tell), which is fixed by 04c
4. 04b_no_accum + 04c - no changes, so 04c is good to go, and it looks like 04b_no_accum can't work without 04c (and viceversa)

#26 - 03/05/2013 08:18 AM - Greg Shah

Go ahead and check these in and distribute them. If Eric has concerns we will address them in a separate distribution. Well done!

#27 - 03/05/2013 08:29 AM - Constantin Asofiei

- File ca_upd20130305b.zip added

Another case of no-reference is:

```
form book.book-title with frame f2.
message book.book-title:label in frame f2.
```

I'm putting it through conversion regression testing now.

#28 - 03/05/2013 08:39 AM - Constantin Asofiei

Greg Shah wrote:

> Go ahead and check these in and distribute them. If Eric has concerns we will address them in a separate distribution. Well done!

0304c.zip was committed as 10240.
0305a.zip was committed as 10241.

#29 - 03/05/2013 08:46 AM - Constantin Asofiei

- File deleted (ca_upd20130305b.zip)

#30 - 03/05/2013 08:47 AM - Constantin Asofiei

- File ca_upd20130305b.zip added

I've fixed the history entry in common-progress.rules.

#31 - 03/05/2013 09:17 AM - Constantin Asofiei

Another side-note for this task: we should identify all missing no-reference cases and code them in common-progress.rules/is_no_reference_item function. Because I have a feeling there are more than we handle now.

#32 - 03/05/2013 10:03 AM - Constantin Asofiei

0305b.zip has passed conversion regression testing.

#33 - 03/05/2013 11:02 AM - Greg Shah

Check it in and distribute.

Should we still test ca_upd20130304b.zip by itself or should we include these other changes?

#34 - 03/05/2013 11:12 AM - Constantin Asofiei

Greg Shah wrote:

> Check it in and distribute.

Committed to bzr revision 10242.

> Should we still test ca_upd20130304b.zip by itself or should we include these other changes?

0304b.zip must be tested together with 0304c.zip, and I will feel better if we test these alone.

Also, as 0305a.zip triggered converted code changes, I suggest to leave it in a separate run, together with 0305b.zip (possible batched with other updates, which do not affect converted code).

#35 - 03/06/2013 08:47 AM - Greg Shah

- % Done changed from 0 to 100

- Status changed from WIP to Closed

#36 - 03/07/2013 03:51 AM - Constantin Asofiei

Another creature of the rabbit hole found at #1985, 120. This fails:

```
def temp-table tt1 field tt1-f1a as int field f1b as int.
def temp-table tt2 field tt1-f2a as int field f2b as int.

find first tt2.
for each tt1 where tt1.f1b = tt2.f2b:
   if tt1 = 0 then message "a".
```

```
end.
```

while this works (note the WHERE clause is not used:

```
def temp-table tt1 field tt1-f1a as int field f1b as int.
def temp-table tt2 field tt1-f2a as int field f2b as int.

find first tt2.
for each tt1:
   if tt1 = 0 then message "a".
end.
```

I guess the tt2 gets promoted to the FOR EACH block, as it was used in the WHERE clause.

#37 - 03/07/2013 04:53 AM - Constantin Asofiei

- File ca_upd20130307a.zip added

I think the sollution was to mark the tables which are weak reference as preferred, when abbreviations are encountered. I'm putting this through conversion regression testing.

#38 - 03/07/2013 05:51 AM - Constantin Asofiei

This has passed conversion regression testing.

#39 - 03/07/2013 11:56 AM - Greg Shah

The change looks fine. Check it in and distribute it.

#40 - 03/07/2013 12:02 PM - Constantin Asofiei

Committed to bzr revision 10262.

#41 - 11/16/2016 10:59 AM - Greg Shah

- Target version changed from Milestone 4 to Conversion Support for Server Features

Files

ca_upd20130304a.zip	312 KB	03/04/2013	Constantin Asofiei
ca_upd20130304b.zip	316 KB	03/04/2013	Constantin Asofiei
ca_upd20130304c.zip	38.2 KB	03/04/2013	Constantin Asofiei
ca_upd20130305a.zip	41.4 KB	03/05/2013	Constantin Asofiei
ca_upd20130305b.zip	40.5 KB	03/05/2013	Constantin Asofiei
ca_upd20130307a.zip	339 KB	03/07/2013	Constantin Asofiei