Database - Bug #2990

inconsequent field names in shared temp-tables

02/09/2016 02:04 PM - Ovidiu Maxiniuc

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

Description

History

#1 - 02/09/2016 02:13 PM - Ovidiu Maxiniuc

I just discovered one of the weirdest things about shred temp-tables in OE. I thought that in #2595 I covered all kind of particular cases, but this is really strange.
It seems that the name of the fields is NOT important, in other words, the same shared temp-tables can name its fields differently in separate source files, the identification is only done by their order and the types of the fields must respect it.
Better with an example:
master.p:

```
DEFINE NEW SHARED TEMP-TABLE stt1
      FIELD f1 AS CHARACTER INIT "6"
      FIELD f2 AS INT INIT 777
      FIELD f3 AS DATE INIT 12/12/12.

CREATE stt1.
DISPLAY stt1.
RELEASE  stt1.

RUN VALUE("servant.p").

MESSAGE "ok M".
```

servant.p:

```
DEFINE SHARED TEMP-TABLE stt1
      FIELD f2 AS CHARACTER INIT '5'
      FIELD f3 AS INT INIT 777
      FIELD f1 AS DATE INIT 12/12/12.

FIND FIRST stt1.
DISPLAY stt1.

MESSAGE "ok S".
```

The above files compiles under OE10.2B and result is:

```
┌──────────────────────────────────┐
│f1                f2 f3           │
│──────── ─────────── ─────────────│
│6                777 12/12/12     │
└──────────────────────────────────┘

┌──────────────────────────────────┐
│f2                f3 f1           │
│──────── ─────────── ─────────────│
│6                777 12/12/12     │
└──────────────────────────────────┘
```

```
ok S
ok M
```

As you can see, the record initialized in master is visible in servant, but the name of the fields are different, I push it a little and used a permutation so a reader will be totally lost when referring to a field of this table.

Well, the big problem is how do we cope with this? The name of the fields are 'hardcoded' in the DMO class. How do implement this: access the same field of a record with a name/method in a 'master' source file and other in 'servant'?

See #3214 for a customer example of this issue.

#2 - 02/10/2016 11:31 AM - Paul E

Well that's weird - defies any reasonable definition of the word "shared".

Is it possible to warn, or even better error and fail, during conversion on encountering this kind of problem? Unless my lack of Progress ABL knowledge is letting me down, I can see no reason why we'd ever want the definition of a shared temp table to differ in any way.

I'm sure you guys can replicate Progress' behaviour here (and probably want to) but from our perspective I think this is an error that we should correct in the Progress source and it's best to know about it as early as possible (e.g. at conversion time rather than at runtime).

If it's hard for you to error out at conversion time then maybe we'll be able to write a custom prolint rule to do this instead.

#3 - 02/11/2016 10:27 AM - Eric Faulhaber

This is a challenging error to detect at conversion time, given the way we currently detect common temp-table definitions, which is largely based on names, data types, and table structure matching. It would require that the matching logic be more fuzzy, which could lead to unintended side effects. I wonder if Progress takes a cue from the temp-table's name, and only allows the leniency if the name is the same across the mismatched tables. If so, that would help narrow the scope.

Anyway, I think the best short-term fix is simply to correct the error in the 4GL code. The obvious problem with that is that we don't know where else in a code base this issue might be lurking.

The better (but more difficult and time consuming) fix would be to warn of the issue during conversion. For our purposes, we wouldn't want to fail the whole conversion as a result, though I understand why this is useful in a CI pipeline environment. We'll have to figure out how to accommodate both.

Ultimately, I would want to detect this situation, issue a warning, and fix it automatically at conversion. This would involve deciding which definition is the authoritative one, fixing the other(s), and fixing up all the code references to the incorrect field in the other(s). That would leave behind rational converted code. The problem is that the algorithm to detect the error must be airtight, so we don't go "fixing" tables which really shouldn't match.

#4 - 02/11/2016 11:18 AM - Paul E

- Assignee set to Paul E

Ok, my concern on this comes from the scale of the unknowns: we've got 13% test coverage and so it's unknown how many other instances of this problem there are.

I'm not sure that I agree with your suggestion of an automatic fix - I think human intervention is required to determine the correct fix.

We'll look into doing something to identify shared temp tables, group them by name and assert that their definitions shouldn't differ.