Conversion Tools - Feature #7178

move the buffer scoping calculations to parse time

03/07/2023 05:24 PM - Greg Shah

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

Description

History

#2 - 03/07/2023 07:41 PM - Greg Shah

When we initially created the parser, SymbolResolver (SR), SchemaDictionary (SD) and the rest of our front end infrastructure, we did not yet have a proper understanding of the nature of record scoping. We did not appreciate the importance or complexity of the record scoping rules until we had already put quite a lot of infrastructure together including a patchwork of approximations built into the parser/SR/SD. Over time we added to those approximations as new issues were found. To date we have always found a way to patch our hacked approximate approach to make things work. With #7125, we've hit some problems that are not easily handled in our approximation approach. The time to layer more patches on top of this mess is over.

Of course, to implement the downstream processing we did have to figure out the rules (most of them anyway) and implement them faithfully. The problem is that these are all handled at the annotations phase (annotations/record_scoping*.rules) and this is way too late to properly resolve all schema references at parse time.

This task is meant to:

1. Update our documentation to match the full rules. We have some documentation and example scope "snippets" here (scroll down to "Record Scopes"). This is out of date, so some rules are not properly defined. On the other hand, it is a pretty good start, especially for the most complex piece which is the weak scope processing and how these scopes combine in weird ways. More recently, we've found some strange behavior with strong scopes which is not documented. Some of that behavior may be implemented at parse time only. We also may have implemented some of the rules in TRPL and/or the BufferScopeWorker without reflecting the rules in the documentation. I propose that we use the buffer scoping testcases being build by Marian's team in #6855 to ensure that we have the proper set of rules defined.

2. Implement these rules in the parser, SR and SD (see Schema Dictionary design). I expect we will remove all the propagation, promotion and other weird patches. I also expect that most (if not all) of the downstream rules will be unnecessary as well. We will still want to retain the BUFFER_SCOPE node and inserting that will be tricky during the parse itself because its position is not known until . We need to decide how to handle it.

There are real benefits to handling it no later than early annotations so that we can see all the results during reporting. Our IDE support will also benefit from early resolution of all of these scopes. I would like to display this information graphically so that the 4GL developer can see it in real time while editing the code.

The shorter term result will be to get the scoping and name resolution to match all known rules exactly.