Database - Feature #1933

Feature # 1581 (Closed): add conversion and runtime support for certain metadata constructs

add conversion support for certain metadata constructs

01/08/2013 12:15 PM - Eric Faulhaber

Status:	Closed	**Start date:**	01/26/2013
Priority:	Normal	**Due date:**	
Assignee:	Eric Faulhaber	**% Done:**	100%
Category:		**Estimated time:**	60.00 hours
Target version:	Conversion Support for Server Features		
billable:	No	**vendor_id:**	GCD

Description

History

#1 - 01/08/2013 12:19 PM - Eric Faulhaber

This task is just about converting these constructs and stubbing out any new runtime methods needed, so that the converted code compiles. Getting the runtime to work is a matter for the parent issue. Please update all history in the parent issue, so we have it in one place.

#2 - 01/14/2013 11:53 AM - Eric Faulhaber

- Assignee changed from Stanislav Lomany to Eric Faulhaber

#3 - 01/31/2013 11:21 PM - Eric Faulhaber

- Status changed from New to WIP

- Start date set to 01/26/2013

#4 - 01/31/2013 11:37 PM - Eric Faulhaber

- % Done changed from 0 to 70

#5 - 02/14/2013 02:05 PM - Eric Faulhaber

- % Done changed from 70 to 100

- File ecf_upd20130212a.zip added

- Status changed from WIP to Closed

Attached update has been regression tested and is committed to bzr revision 10170.

#6 - 02/28/2013 02:43 PM - Constantin Asofiei

The metadata tables are emitted to the dmo._meta package, but all buffers referring meta tables are still converted to refer DMOs from the dmo.dbname package. When used, compile errors will appear, as the dmo._meta package is not imported.

#7 - 02/28/2013 02:49 PM - Eric Faulhaber

Ah, good point. So, we need to emit an import statement for ...dmo._meta.*, right? Are you taking this, or do you want me to?

BTW, I'm thinking of moving the metadata DMOs into a com.goldencode.p2j.persist.meta package soon, since we will need to access these to service many of the converted 4gl attributes from the runtime, so we will need to revisit this...

#8 - 02/28/2013 03:33 PM - Constantin Asofiei

Correct, we need the import.

Ovidiu was looking at this at the end of the day today, I guess he will continue tomorrow.

#9 - 03/01/2013 03:28 AM - Ovidiu Maxiniuc

Eric,
I exported a sample _File structure (.df) and data (.d) from my test database but, for some unknown (to me) cause, when doing the conversion, my File interface is generated in the package

```
package com.goldencode.testcases.dmo.p2j_test
```

as a normal table, instead of ...dmo._meta, even though I activated it the meta-table in the p2j.cfg.xml:

```
   <metadata name="standard">
      <table name="_file" />
      ...
   </metadata>
```

After a short discussion with Constantin, the conclusion was that I need a special configured project or better, to let you continue with this issue.

#10 - 03/01/2013 07:50 AM - Eric Faulhaber

Ovidiu Maxiniuc wrote:

> I exported a sample _File structure (.df) and data (.d) from my test database ...

I can take this one over, but for future reference, you do not need to export anything. There already should be a file named standard.df (or similar) in your project. This is where all the metadata tables are defined. If you just change the p2j.cfg.xml file as you noted above, the File DMO interface and FileImpl DMO implementation class should be generated during the m0 portion of the conversion, and they should have the ...dmo._meta.* package statement.

#11 - 03/01/2013 10:13 AM - Ovidiu Maxiniuc

I will go on with this issue, as I am now able to see what's wrong using my project.

#12 - 03/02/2013 12:42 PM - Ovidiu Maxiniuc

For identifying the correct package to be imported I went back on the annotation paths for pkgname, bufname, schemaname and dbname all the way back to initial ast creation of BUFFER_SCOPE. This lead to record rule of progress.g where the initial ast is created and annotated. The values are lookup-ed from SymbolResolver. And there they are added from SchemaLoader.postProcessImport().
The important thing here is the merging of the meta tables (loaded from standard.df) with current database. Something is not right there:

- 4GL has a meta set of tables associated with each database and from the user point of view the tables are just there, the only difference is that their name start with underscore. (I even exported some meta-tables form 4GL and they were dumped into the same df file as the rest of the tables of the database.) So it makes sense to merge them into a single P2J database. However, doing this the meta attribute is lost, and the correct package cannot be inferred when generating java code. A fix here is rather simple: just need to flag (annotate) the meta tables before merging, maybe just at parsing time and use this later when setting the needed pkgname.
- Another issue is name collision. If the client application has a table with the same name (but without the underscore) the generated dmo will have the same name and the reference will be ambiguous in generated code. As a solution we could name the meta dmo-s using 4GL convention (starting with starting underscore). Maybe it's not the best solution, just an idea, as using other prefixes could also lead to collisions.
- If two or more databases are used in the application, each of them will have a different meta set of tables. But there is only one package _meta so they will collide. A meta package should be created for each database and positioned inside database's import package. Or, if using the solution from the above point, the dmo-s can be directly mixed into the same package.

#13 - 03/04/2013 11:07 AM - Eric Faulhaber

Ovidiu Maxiniuc wrote:

> 4GL has a meta set of tables associated with each database and from the user point of view the tables are just there, the only difference is that their name start with underscore. (I even exported some meta-tables form 4GL and they were dumped into the same df file as the rest of the tables of the database.) So it makes sense to merge them into a single P2J database. However, doing this the meta attribute is lost, and the correct package cannot be inferred when generating java code. A fix here is rather simple: just need to flag (annotate) the meta tables before merging, maybe just at parsing time and use this later when setting the needed pkgname.

I think this is probably best done in P2OLookup (in the schema package), such that P2OLookup.javaPackage(String historical) returns the appropriate ...dmo._meta package string if the historical parameter is included in a (new) set of meta-table names collected during initialization. If not, it delegates to the getJavaPackage method as today. This will require some code to be added to collect the set of fully-qualified, historical names of all metadata tables in use (using all customer database names as the database qualifier).

> Another issue is name collision. If the client application has a table with the same name (but without the underscore) the generated dmo will have the same name and the reference will be ambiguous in generated code. As a solution we could name the meta dmo-s using 4GL convention (starting with starting underscore). Maybe it's not the best solution, just an idea, as using other prefixes could also lead to collisions.

I thought of this, but I didn't like the idea of starting a Java class name with an underscore. I figured we could just deal with collisions using schema hints to change conversion of customer tables if necessary. One thing I don't want to do, though, it to have the metadata DMO names vary by project, since there will be runtime code which relies on well-known metadata DMO names.

> If two or more databases are used in the application, each of them will have a different meta set of tables. But there is only one package _meta so they will collide. A meta package should be created for each database and positioned inside database's import package. Or, if using the solution from the above point, the dmo-s can be directly mixed into the same package.

I don't understand this point. The list of metadata tables in p2j.cfg.xml was meant to be the union of all metadata tables in use across multiple

databases. The intention was to have only one ...dmo._meta package. Where would the collision be?

#14 - 03/04/2013 01:04 PM - Ovidiu Maxiniuc

> I think this is probably best done in P2OLookup (in the schema package), such that P2OLookup.javaPackage(String historical) returns the appropriate ...dmo._meta package string if the historical parameter is included in a (new) set of meta-table names collected during initialization. If not, it delegates to the getJavaPackage method as today. This will require some code to be added to collect the set of fully-qualified, historical names of all metadata tables in use (using all customer database names as the database qualifier).

You mean that we should use P2OLookup instead of the current SymbolResolver calls ? I understand that P2OLookup will be used later in the process, just when generating java code. On the other hand, the ast look like this:

```
    <ast col="0" id="558345748584" line="0" text="p2j_test._file_p2j_test._file" type="BUFFER_SCOPE">
      <annotation datatype="java.lang.String" key="bufname" value="p2j_test._file"/>
      <annotation datatype="java.lang.String" key="schemaname" value="p2j_test._file"/>
      <annotation datatype="java.lang.String" key="dbname" value="p2j_test"/>
```

Is this normal ? I incline to believe so, after merging from postProcessImport() from SchemaLoader.

> I thought of this, but I didn't like the idea of starting a Java class name with an underscore. I figured we could just deal with collisions using schema hints to change conversion of customer tables if necessary. One thing I don't want to do, though, it to have the metadata DMO names vary by project, since there will be runtime code which relies on well-known metadata DMO names.

If the metadata is processed before before other tables, they will use the same names as they are not taken by other permanent tables. At this moment, having a permanent table named File will confuse P2J. A dmo with the same name is created and even if the imports would be correct, the class names must be fully specified in generated code as the java compiler will complain about the ambiguity.

> I don't understand this point. The list of metadata tables in p2j.cfg.xml was meant to be the union of all metadata tables in use across multiple databases. The intention was to have only one ...dmo._meta package. Where would the collision be?

I cannot tell at this moment if metadata contains the same kind of information across databases. For example, if the application uses two different databases, but exported from different Progress versions. I noted that the standard.df found in my project (I believe it's a 9.x export) is very different than the _meta.df file I exported from windev (which is version 10.x)

#15 - 03/04/2013 03:12 PM - Eric Faulhaber

Ovidiu Maxiniuc wrote:

> You mean that we should use P2OLookup instead of the current SymbolResolver calls ?

No. Leave the parser alone, it is generating the correct annotations.

> I understand that P2OLookup will be used later in the process, just when generating java code.

Yes, that's right, and that's where we should address this problem, not in the parser or SymbolResolver. I'm suggesting we change the way the P2OLookup.javaPackageName returns package names, so that when it is called by P2OAccessWorker to provide a package name to be stored in a "pkgname" annotation in a buffer or buffer scope node representing a metadata table, it reflects the fact that the buffer's backing DMO is in the ...dmo._meta package, not the ...dmo.<appname> package. Later, when the "pkgname" annotation is retrieved during core conversion, it will have the correct package name for a buffer.

> On the other hand, the ast look like this:
> [...]
> Is this normal ? I incline to believe so, after merging from postProcessImport() from SchemaLoader.

Yes, this is normal and correct. The Progress code treats the metadata tables as part of the current database, so we do, too.

> If the metadata is processed before before other tables, they will use the same names as they are not taken by other permanent tables. At this moment, having a permanent table named File will confuse P2J. A dmo with the same name is created and even if the imports would be correct, the class names must be fully specified in generated code as the java compiler will complain about the ambiguity.

I understand the potential for conflict. We have several architected solutions for database table name conflicts. Please see the Conversion Hints chapter and the Other Customizations chapter of the P2J Conversion Reference. I am saying we should use schema hints to resolve any name collisions that come up. For your example, we might use something like this:

```
<hints>
   <schema>
      <table name="file">
         <phrase match="file" replace="something else" />
      </table>
   </schema>
</hints>
```

This would cause the permanent table file to convert to SomethingElse (DMO name) and something_else (table name).

> I cannot tell at this moment if metadata contains the same kind of information across databases. For example, if the application uses two different databases, but exported from different Progress versions. I noted that the standard.df found in my project (I believe it's a 9.x export) is very different than the _meta.df file I exported from windev (which is version 10.x)

At this point, I am OK limiting our conversion support to a single version of Progress per project.

Bottom line

This fix requires a change to one file, P2OLookup.java. Add a private instance variable (say, historicalMetaTables) which is a Set<String>. Add a boolean meta parameter to initialize(String, Aast), and refactor initialize(String) to call that method with meta set to true when processing the metadata AST. In initialize(String, Aast, boolean), collect the fully qualified, historical table names in historicalMetaTables if meta == true. In P2OLookup.javaPackageName, return the ...dmo._meta name if the passed name is present in the P2OLookup's set of names, else return the regular package name.

With this change, the correct import statement should naturally be included into any converted program with a reference to a metadata table.

#16 - 03/05/2013 07:52 AM - Ovidiu Maxiniuc

- File om_upd20130305a.zip added

Fixed java package for metadata tables.

Just a question. I know that the problem is solvable by using hints but if a name conflict occurs between a meta table and a normal one at this moment there is no warning generated, except for those of java compiler. Shouldn't we use an automated postfixed name change if some name is already taken? Or at least print some warning/errors in this case ?

#17 - 03/05/2013 09:13 AM - Eric Faulhaber

Ovidiu Maxiniuc wrote:

> Shouldn't we use an automated postfixed name change if some name is already taken? Or at least print some warning/errors in this case ?

I don't want to automatically change the table name in this case; I'd rather know about the conflict and make the change manually. So, I think the warning is a good idea to find out about the problem before the compiler lets us know.

#18 - 03/05/2013 10:04 AM - Ovidiu Maxiniuc

Should I add the verification? What kind should it be? Just a System.err/out.print() or persisted to logging file ?

At this moment I "know" how to detect if these collisions occurs within the same schema, checking for collisions between separate schemas need more code than a simple content check.

#19 - 03/05/2013 10:11 AM - Eric Faulhaber

I only want you to add this feature if it is very fast to implement, since this is not in the critical path. Otherwise, just add the requirement to #1581 and move on to other M4 issues.

If in a rule-set, use printfln, if in java then yes, use System.out.println.

Don't worry about cross-schema collisions, this will take longer and is not in the critical path. Please note this limitation in #1581.

#20 - 03/05/2013 10:56 AM - Ovidiu Maxiniuc

- File om_upd20130305b.zip added

This update adds a message to standard console when a duplicate dmo name is detected.

#21 - 03/05/2013 10:15 PM - Eric Faulhaber

Code review 20130305a:

Changes look good.

Code review 20130305b:

The change is fine, but a slightly tighter way would have been:

```
if (!javaNames.add(ifaceName))
{
   System.out.println("Duplicate dmo interface name detected: [" + ifaceName + "]");
}
```

instead of:

```
if (javaNames.contains(ifaceName))
{
   System.out.println("Duplicate dmo interface name detected: [" + ifaceName + "]");
}
else
{
   javaNames.add(ifaceName);
}
```

I am putting both changes (as is) into conversion regression testing.

#22 - 03/05/2013 11:41 PM - Eric Faulhaber

Conversion regression testing was successful. Please commit and distribute the 20130305b update.

#23 - 11/16/2016 11:07 AM - Greg Shah

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

Files

ecf_upd20130212a.zip	321 KB	02/14/2013	Eric Faulhaber
om_upd20130305a.zip	9.86 KB	03/05/2013	Ovidiu Maxiniuc
om_upd20130305b.zip	9.93 KB	03/05/2013	Ovidiu Maxiniuc