Database - Feature #2120

Feature # 1651 (Closed): add support for dynamic database features

Feature # 1653 (Closed): add conversion and runtime support for dynamically prepared queries

add runtime support for dynamically prepared queries

04/09/2013 01:18 PM - Eric Faulhaber

Status:	Closed	**Start date:**	04/02/2013
Priority:	Normal	**Due date:**	05/27/2013
Assignee:	Constantin Asofiei	**% Done:**	100%
Category:		**Estimated time:**	100.00 hours
Target version:	Runtime Support for Server Features		
billable:	No	**vendor_id:**	GCD

Description

History

#1 - 04/09/2013 01:29 PM - Eric Faulhaber

- Estimated time changed from 80.00 to 100.00

A prototype will integrate limited conversion support into the runtime environment, such that given a set of buffers and a 4GL WHERE clause referencing static tables, we can produce and use a compatible Java query. This will require:

- moving static schema dictionary intermediate form resources (*.dict) files into a runtime-accessible location (an application jar file);
- preparing a SchemaDictionary instance for parsing the WHERE clause;
- lexing and parsing the WHERE clause into an AST;
- processing the AST using the pattern engine with the minimum necessary subset of existing TRPL rulesets through the annotations phase (certain features will have to be adjusted, such as persisting to file and writing messages to the console);
- extracting the necessary data from the annotated AST (e.g., HQL WHERE clause and ORDER BY clause);
- preparing the appropriate runtime, Java query using these data within a QueryWrapper (or subclass);
- opening the query and executing subsequent instructions (e.g., GET-FIRST/NEXT).

It does not include (but its design should not preclude):

- complex WHERE clauses using natural joins (i.e., the OF keyword) or dynamic temp-tables;
- optimization of conversion resources into a queued service (i.e., to prevent one instance of SchemaDictionary in memory per session);
- other performance/resource optimizations.

#2 - 04/18/2013 11:07 AM - Eric Faulhaber

- Status changed from New to WIP

- Start date set to 04/02/2013

#3 - 04/25/2013 11:34 AM - Eric Faulhaber

- Due date set to 05/27/2013

#4 - 10/04/2013 11:17 AM - Eric Faulhaber

- Assignee changed from Eric Faulhaber to Constantin Asofiei

#5 - 10/08/2013 08:29 AM - Eric Faulhaber

I wrote:

> optimization of conversion resources into a queued service (i.e., to prevent one instance of SchemaDictionary in memory per session)

...as an idea for the production version of this facility.

However, after some discussion with Greg, it seems that making the SchemaDictionary threadsafe, and just having one instance in memory is a preferred approach.

It already is safe to run separate instances of ProgressParser on different threads. However, some work will be necessary to make SchemaDictionary threadsafe. The idea is to share the static, permanent table data it holds and to make any session-specific state (e.g., dynamic temp-tables and buffers, or which static temp-tables are in scope when a particular program is running) context local.

However, I haven't been in this code for a long time, so this approach needs a reality check.

#6 - 10/08/2013 01:41 PM - Constantin Asofiei

Some initial thoughts on this: the documentation (and practice) shows that for a prepared query string:

1. it follows the syntax of the OPEN QUERY clause, without the OPEN QUERY prefix
2. reference to non-buffer data (i.e. widgets, variables) is not allowed
3. reference to other buffers (not registered with the query) is not allowed
4. for all registered buffers, a {FOR | PRESELECT} EACH (for the first buffer) or FOR {EACH | FIRST | LAST} (for joined buffers) must be specified

What are the conclusions for all of this? We can't parse just the WHERE clause. We need to parse the entire query string, as we need to determine:

1. the query equivalent in P2J (AdaptiveQuery, PreselectQuery, etc)
2. where clause
3. index

This info can't be easily extracted from a ProgressAst. Is easier to go all the way and generate the entire final JavaAast, which can be walked and initialize the query. More, I don't want to build the ProgressAst by hand (as I think we will end up duplicating complexity from the TRPL rules inside Java code); instead, what I'm thinking is to build a 4GL code snippet (which will be used as input) and which will have this structure:

1. temp-table definition (when a buffer references the temp-table)
2. maybe define an explicit buffer for each buffer used by the query. But this will need to be mapped with the buffers used in the WHERE clause; I think this is will not be too difficult, as if we make the resulted HQL query generic, will help optimization. We can cache the instantiated query (to some extent) for a certain (table list, prepare-query string) pair (we can't use the buffer names, as a buffer might reference a certain table now and different one later). If the resulted WHERE clause uses static buffer names, we can easily replace them with the runtime buffer names.
3. the OPEN QUERY command, like OPEN QUERY <query-prepare>..

Ideally, we will be able to dynamically add in the ProgressAst the BUFFER_SCOPE nodes for each buffer and also dynamically build the associated .dict definition for a temp table, to let only the OPEN QUERY command to be parsed, but I'm not yet convinced this will not be very complex (or even possible), as the query parsing requires the SymbolResolver to be initialized with the temp-table data, before the OPEN QUERY is parsed.

#7 - 10/08/2013 01:53 PM - Greg Shah

Interesting results. These are good limitations as they greatly reduce the possible functionality.

> temp-table definition (when a buffer references the temp-table)

I hope you can avoid this.

If the TT is made using CREATE TEMP-TABLE, then during our runtime processing of the definition, we should be calling methods in the SchemaDictionary to add TT configuration that represents this table. This is what we do for DEFINE TEMP-TABLE in the parser. Perhaps you need slightly different methods, but the pieces should already be there.

If the TT is static (via DEFINE TEMP-TABLE/WORK-TABLE), then we should force that the matching configuration to be loaded into the SchemaDictionary. This is information we should save at conversion-time. I think we can include it in the application jar file and add a process to load it. OR we can leverage the XML structure Eric is building for permanent table metadata.

But if the SchemaDictionary is properly configured, it seems that we shouldn't need to fake out the definitions as 4GL code.

#8 - 10/08/2013 02:10 PM - Constantin Asofiei

Don't know why I forgot the SymbolResolver is backed by the SchemaDictionary for temp-table stuff; with this in mind, I should be able to manage to avoid faking out the definitions as 4GL code.

> If the TT is made using CREATE TEMP-TABLE, then during our runtime processing of the definition, we should be calling methods in the SchemaDictionary to add TT configuration that represents this table.

We will need to sync it with the resource's scoping.

> If the TT is static (via DEFINE TEMP-TABLE/WORK-TABLE), then we should force that the matching configuration to be loaded into the SchemaDictionary. This is information we should save at conversion-time. I think we can include it in the application jar file and add a process to load it. OR we can leverage the XML structure Eric is building for permanent table metadata.

The temp-table name is not unique throught the application. More, the temp-table definition might come from a completely different file... when we define a temp table in P2J, we will need to resolve its associated legacy configuration (.dict file or anything else), based on the program name where the definition takes place. We can register it with the SchemaDictionary upfront, but we will a scoping mechanism along with it, as at a certain point only certain temp-tables are visible.

#9 - 10/16/2013 05:48 AM - Constantin Asofiei

Related to note 8 - see the discussion in notes 20-36 in #1654. Summary: TableMapper was added to map legacy names for temp-tables.

Next on the table is how the dynamic queries work. In 4GL, a static query can have its predicate changed, via QUERY-PREPARE; beside this, some state can not be changed (i.e. buffers can't be added for a static query), but other state can be interrogated (i.e. INDEX-INFORMATION works for static queries too). With this in mind, the implementation of QUERY-PREPARE method (which changes the underlying P2J query) must be at the QueryWrapper class. Some of the dynamic query-related APIs need to be implemented at QueryWrapper, and not at AbstractQuery. AbstractQuery should throw a RuntimeException if some code finds its way there.

The conclusion is that for us, QueryWrapper is the most appropriate to mark as the legacy resource. A handle should never reference an AbstractQuery sub-class, only a QueryWrapper can be referenced.

#10 - 10/17/2013 08:12 AM - Constantin Asofiei

About extracting the query instance from a generated JavaAST. The current approach is to parse a 4GL code like OPEN QUERY q ..., generate the JavaAst and extract/create the P2JQuery instance from it. This extraction needs to work like a "just-in-time" compiler and there are two approaches:

1. walk the JavaAst and instantiate the query, add components, etc (this will be pretty complex, as there are lots of cases which will need to be treated). Consider that the following code is a valid prepare query:

```
def temp-table tt1 field f1 as int field f2 as int extent 2.

create tt1.
tt1.f1 = 5.
tt1.f2[1] = 1.
tt1.f2[2] = 2.

def var hq as handle.
create query hq.
hq:add-buffer(buffer tt1:handle).
hq:add-buffer(buffer book:handle).
hq:query-prepare("for each tt1, each book where book.book-id = tt1.f1 and tt1.f2[1] > 0").
hq:query-open().

hq:get-first().

message tt1.f1 book.book-id book.book-title.
```

   During conversion of a open query for each tt1, each book where book.book-id = tt1.f1 and tt1.f2[1] > 0 this will result in client-side where clauses, multiple components, etc.
2. The second approach would be to alter the generated JavaAST so that this code is produced (an example for the for each tt1, each book where book.book-id = tt1.f1 and tt1.f2[1] > 0 predicate:

```
public class InMemQuery
{
   /* all buffer fields are uninitialized and made public */
   public Book.Buf book = null;

   public TempRecord1.Buf tt1 = null;

   /* the query field is moved to the top-level class */
   CompoundQuery query0 = null;

   WhereExpression whereExpr0 = new WhereExpression()
   {
      public logical evaluate(final BaseDataType[] args)
      {
         return isGreaterThan(tt1.getF2(0), 0);
      }
   };

   /* a getter for the generated query is added */
```

```
   public P2JQuery getQuery()
   {
      return query0;
   }

   public void execute()
   {
      // the RecordBuffer.openScope and the query0.open() calls are dropped, and all the code between thes
e two calls are moved to the execute() method
      query0 = new CompoundQuery(true, false);
      query0.addComponent(new AdaptiveQuery(tt1, (String) null, null, "tt1.id asc"));
      RecordBuffer.prepare(book);
      query0.addComponent(new AdaptiveQuery(book, "book.bookId = ?", whereExpr0, "book.bookId asc", new Ob
ject[]
      {
         new FieldReference(tt1, "f1")
      }));
   }
}
```

Thus, we can use the same approach as with the in-memory DMOs: the source code is compiled in memory, added to the InMemoryClassLoader and this new class is used to instantiate the query using reflection, following these steps:
- create a new instance of this class
- as the names of the instance fields referencing the buffers have the same names as the passed buffers, set them accordingly
- call execute() on this instance to initialize the query
- call getQuery() on this instance to extract the created query.

#11 - 10/17/2013 08:54 AM - Greg Shah

The 2nd approach (in-memory compilation of real java code, like with the dynamic DMOs) is the one that I believe is the best option. It should be simpler (less work). It is also less risky because we don't have to create an interpreter which is essentially a parallel implementation of the some significant runtime functionality. I think the time to generate and compile the Java code will be a minimal cost to pay.

#12 - 10/17/2013 11:42 AM - Constantin Asofiei

Stanislav, is there a specific reason why you chose to explicitly store objects rather than build something generic in persist() and persistJavaFile()? I ask this because to me it looks cleaner to catch the Configuration.isRuntimeConfig case in these methods, rather than explicitly calling them in various places throughout the TRPL code. What I'm thinking is something like this:

1. always use the fully-qualified converted class name as the key
2. use categories based on the AST class, to differentiate between i.e. JavaAST and ProgressAST
3. persist() and persistJavaFile() will check for in-memory conversion case and if so, it will store the AST in the required category, depending on the AST's class.

#13 - 10/17/2013 01:08 PM - Stanislav Lomany

Stanislav, is there a specific reason why you chose to explicitly store objects rather than build something generic in persist() and persistJavaFile()?

At that point I tried to make a generic solution that works with abstract in-memory objects and require minimal changes in the existing conversion rules. Note that we can have xml files and may have additional steps for building file path in the case of persistent files.

#14 - 10/20/2013 02:31 PM - Constantin Asofiei

Some notes on my current status (Stanislav/Greg/Eric, please check the questions):

1. following test does convert and produces valid output

```
def temp-table tt1 field f1 as int.

create tt1.
tt1.f1 = 10.
create tt1.
tt1.f1 = 20.

open query q for each tt1 where tt1.f1 < 20.
get first q.
message tt1.f1.

release tt1.

def var h as handle.

create query h.
h:add-buffer(buffer tt1:handle).
h:query-prepare("for each tt1 where tt1.f1 < 20").
h:query-open().
h:get-first().

message tt1.f1.
```

2. to some extent, static queries can be made dynamic (i.e.they can have their predicate changed at runtime). I will implement this too, if is not too much work.
3. currently, the entire annotations/base_structure/core_conversion rulesets are ran, but I will try to limit this to the query-related rulesets. Greg: is there a way to conditionally include a rule-set? I'm thinking of doing something like this, in i.e. annotations.xml:

```
   <!-- main processing (once per tree) -->

   <!-- must be executed first, to make sure ASTs are not removed, altered -->
   <rule-set condition="!isRuntimeQuery()" name="annotations/collect_published_events" />

   <rule-set condition="!isRuntimeQuery()" name="annotations/process_reachable_refs" />
```

   but I'm not sure how easy is to integrate it. After looking through the query-related conversion code, I'm a little reluctant in creating rulesets for the annotation/core-conversion of dyn queries. IMO it will be cleaner to conditionally include rulesets. Instead of using expressions for the condition attribute, we can define some static flags which can be checked from the Configuration object (I think this will be easier to do). We should allow more than one flag to be set for a rule-set, too.
4. had to build a InMemoryRegistryPlugin which is hooked up as the default AstManagerPlugin for dynamic queries. This acts like an in-memory storage for the created ASTs. The generic storage mechanism is still used to retrieve the java code from brew.xml, so that is still needed.
5. the support for the DYNAMIC attribute needs to be fixed, as this attribute is for more than one resource (currently is added in various interfaces). Stanislav: have you touched this? If not, I will work on it.

6. the dyn query implementation relies on the BUFFER:TABLE-HANDLE attribute to work for both static and dynamic temp-tables. Stanislav: is this going to be supported by the first version of the dynamic buffer implementation?
7. need to add index support at the .dict/.p2o integration for temp tables. But for this, I will need the legacy name/definition of the index. As Java 7 does not allow repeating annotations (Java 8 adds this), we can add something like this, at the DMO implementation class:

```
@LegacyIndexes({@LegacyIndex(name = "idx1", primary = true,
                             fields = @LegacyIndexFields({@LegacyIndexField(name = "f1", asc = false),
                                                          @LegacyIndexField(name = "f2", asc = false)})),
                @LegacyIndex(name = "idx2", unique = true,
                             fields = @LegacyIndexFields({@LegacyIndexField(name = "f3", asc = true),
                                                          @LegacyIndexField(name = "f4", asc = false)}))})
public class TempRecord
{
...
}
```

where the annotation classes look like:

```
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
public @interface LegacyIndexes
{
   LegacyIndex[] value();
}

@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
public @interface LegacyIndex
{
   String name();

   boolean unique() default false;

   boolean primary() default false;

   boolean word() default false;

   LegacyIndexFields fields();
}

@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
public @interface LegacyIndexFields
{
   LegacyIndexField[] value();
}

@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
public @interface LegacyIndexField
{
   String name();

   boolean asc() default true;
}
```

Eric: please let me know what you think.
8. need to check how DB aliases are used in prepare strings
9. need to check if my current SchemaDictionary can work easily with runtime-connected DBs
10. finish the others query attributes/methods

#15 - 10/20/2013 09:03 PM - Eric Faulhaber

Constantin Asofiei wrote:

> following test does convert and produces valid output
> [...]

Very cool!

> need to add index support at the .dict/.p2o integration for temp tables. But for this, I will need the legacy name/definition of the index. As Java 7 does not allow repeating annotations (Java 8 adds this), we can add something like this, at the DMO implementation class:
> [...]
> where the annotation classes look like:
> [...]
> Eric: please let me know what you think.

This looks like it will do the job, but note that we already have a lot of knowledge of index information in the runtime via the IndexHelper class. It may not be served exactly the way you need it, but I'm OK with changes to the API if it's close enough. You would need to use the field-level annotations you have added to the DMOs to replace the property names with their original, legacy counterparts at a minimum. Have a look and see if it meets your needs. If not, we can go with your proposed approach, but I'm concerned about maintaining this information in multiple places.

#16 - 10/21/2013 02:21 AM - Stanislav Lomany

> 5. the support for the DYNAMIC attribute needs to be fixed, as this attribute is for more than one resource (currently is added in various interfaces). Stanislav: have you touched this? If not, I will work on it.

No, I haven't touched this.

> 6. the dyn query implementation relies on the BUFFER:TABLE-HANDLE attribute to work for both static and dynamic temp-tables. Stanislav: is this going to be supported by the first version of the dynamic buffer implementation?

AFAIK, we do not support temp-table handles, so I guess it will not come with the very first version.

#17 - 10/21/2013 02:30 AM - Constantin Asofiei

> AFAIK, we do not support temp-table handles, so I guess it will not come with the very first version.

The dynamic temp-table stuff you've added implements a dynamic TEMP-TABLE resource. But I need the equivalent for static temp-tables, as I don't want to keep the legacy temp-table mappings by temp-table name (as in 4GL they seem to be referenced internally by the UNIQUE-ID attribute). I don't need full support, I need just the TEMP-TABLE:UNIQUE-ID attribute and the TempTableBuilder.persistentClass to be implemented for static temp-tables.

#18 - 10/21/2013 07:35 AM - Constantin Asofiei

- File delete_static_resource.p added

- File ca_upd20131021c.zip added

Adds DYNANIC attribute support; main rules are:

1. static resources can't be deleted
2. for static widget case, an error is raised
3. for static temp-table, query and buffer, no error is raised and no error msg is recorded.

I've added support for all resources implementing this attribute, with these notes:

- a static widget can't be deleted and error is raised
- temp-table:dynamic support is only for dynamic temp-tables
- I've fixed conversion of TEMP-TABLE tt1:HANDLE to TempTableBuilder.asHandle(Temporary)
- buffer:dynamic support needs to be finished
- query:dynamic will be treated in a later update, together with the dynamic query stuff

Please review. I'm running conversion testing now.

#19 - 10/21/2013 07:50 AM - Constantin Asofiei

Eric Faulhaber wrote:

> ... we already have a lot of knowledge of index information in the runtime via the IndexHelper class. ... Have a look and see if it meets your needs.

Actually, I think the DMOIndex.databaseIndexes is the right API for me; this will give me a list of P2jIndex instances with the full index definition. The only thing missing is the legacy index name. For this, I will need annotations like:

```
@LegacyTable(name = "tt1")
@LegacyIndexes({@LegacyIndex(name = "idx-1", converted = "idx1"),
                @LegacyIndex(name = "idx-2", converted = "idx2")})
public class TempRecord1
{
...
}
```

and only the LegacyIndexes and LegacyIndex annotations will be needed:

```
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
public @interface LegacyIndexes
{
   LegacyIndex[] value();
}

package com.goldencode.p2j.persist;

import java.lang.annotation.*;

@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
public @interface LegacyIndex
{
   String converted();

   String name();
}
```

#20 - 10/21/2013 08:05 AM - Eric Faulhaber

I should have thought of this earlier: I am regression testing metadata support (#2115), which includes the _Index table and _Index-Field table. I am currently regression testing the base support; next I will add a MetaSchema interface, which is an internal API to support attributes and other cases where we need legacy information. I can add an API to the MetaSchema interface to give you exactly what you need, without having to maintain those annotations separately. What would these API signatures look like for your needs?

#21 - 10/21/2013 08:07 AM - Constantin Asofiei

Eric, the same problem with the legacy temp-table name: temp-table information is not in the legacy metadata tables; only the permanent tables are supported there. Unless there is something I'm missing, I don't think we can get away from using DMO impl class annotations with the legacy index names.

#22 - 10/21/2013 08:13 AM - Eric Faulhaber

Yes, good point. Sorry, haven't had my morning coffee yet. <g>

#23 - 10/21/2013 08:38 AM - Greg Shah

> currently, the entire annotations/base_structure/core_conversion rulesets are ran, but I will try to limit this to the query-related rulesets. Greg: is there a way to conditionally include a rule-set? I'm thinking of doing something like this, in i.e. annotations.xml:

```
   <!-- main processing (once per tree) -->

   <!-- must be executed first, to make sure ASTs are not removed, altered -->
   <rule-set condition="!isRuntimeQuery()" name="annotations/collect_published_events" />

   <rule-set condition="!isRuntimeQuery()" name="annotations/process_reachable_refs" />
```

> but I'm not sure how easy is to integrate it. After looking through the query-related conversion code, I'm a little reluctant in creating rulesets for the annotation/core-conversion of dyn queries. IMO it will be cleaner to conditionally include rulesets. Instead of using expressions for the condition attribute, we can define some static flags which can be checked from the Configuration object (I think this will be easier to do). We should allow more than one flag to be set for a rule-set, too.

Sure, you can add set of flags for conditional loading. I prefer not to use an embedded expression. But flags are OK. We already do have something similar in our attribute optional="true". See the ruleSet() method in ConfigLoader where we process ATTR_OPT. The idea here would be the following:

- Mark the rule-sets that you need to include with a flag like load-condition="runtime-query".
- This value can be a comma-separated list of flags, so that we can create different "sets" to be loaded.
- By default, we process all rule-sets without honoring the load-condition processing.
- When in runtime query mode, we update the Configuration object to turn on the load limit mode and explicitly enable some list of flags.
- ConfigLoader.ruleSet() is modified to process this flag and when the load limit mode is enabled, only those rule-sets that match one of the enabled flags will be added to the maps of rule-sets. All other rule-sets are silently ignored.

#24 - 10/21/2013 09:25 AM - Constantin Asofiei

- File ca_upd20131021d.zip added

Attached update adds legacy index annotations at the DMO implementation classes. This and the info provided by the DMOIndex.databaseIndexes will be used to create the legacy index definition at the .p2o file level.

Next on the queue is executing rule-sets conditionally. The plan is to finish this too (maybe today) and regression test all these (DYNAMIC attribute, legacy indexes and conditional rule-sets) together and release them once they pass regression and review.

#25 - 10/21/2013 04:01 PM - Constantin Asofiei

- File ca_upd20131021g.zip added

- File ca_upd20131021f.zip added

1021g.zip is replacement for 1021d.zip (legacy index annotations); changes:

- fix bug to ignore synthetic indexes
- TableMapper APIs which resolve field/index names will return null if no mapping is found (to provide a way to disambiguate between index columns which don't have legacy counterparts)

1021f.zip contains the changes for the conditional rule-sets. The changes are pretty simple, check the load-condition attribute against a set of registered flags, if conditional rule-sets is enabled.

These and the 1021c.zip update for the DYNAMIC attribute support are going through regression testing now.

#26 - 10/22/2013 06:33 AM - Stanislav Lomany

> I need just the TEMP-TABLE:UNIQUE-ID attribute and the TempTableBuilder.persistentClass to be implemented for static temp-tables.

Constantin, in order to generate UNIQUE-ID for static temp-tables, their creation should be tracked at runtime. Do we already have a facility impelmented for that?

#27 - 10/22/2013 08:06 AM - Constantin Asofiei

Stanislav Lomany wrote:

> Constantin, in order to generate UNIQUE-ID for static temp-tables, their creation should be tracked at runtime. Do we already have a facility impelmented for that?

The TEMP-TABLE resource should be created/dropped when the temp-table is created/dropped in P2J; there are no notifiers, for mapping I've just added TableMapper.map/removeTemporaryTable(s) in TemporaryBuffer$Context.dropTable, TemporarBuffer$Context.createTable and

TemporaryBuffer$Context.dropAllTables.

Btw, I see that I've added a TableMapper.removeTemporaryTable call in TemporaryBuffer.dropTable - but I think this is incorrect, as the mappings need to be removed when the temp-table is actually dropped. I'll fix this.

#28 - 10/22/2013 08:16 AM - Constantin Asofiei

1021g.zip (legacy index annotations), 1021f.zip (conditional rule-sets) and 1021c.zip (DYNAMIC attribute support) have passed conversion and runtime testing.

#29 - 10/22/2013 08:38 AM - Greg Shah

Code Review 1021c

1. It seems like the methods_attributes.rules change should have hwrap as DynamicResource not Dynamicable. But then again, the method name is unwrapDynamicable(). This seems broken.

2. TempTableBuilder._dynamic() is hard coded to true. Why not just return the value of the dynamic data member?

#30 - 10/22/2013 08:40 AM - Greg Shah

Code Review 1021f

I am fine with the change to brew.xml. Eric will have to review the rest.

#31 - 10/22/2013 08:51 AM - Greg Shah

Code Review 1021g

1. I prefer the enable/disable of conditional rule-set processing to be done at the PatternEngine level. That would allow different instances of the pattern engine to be configured differently within the same JVM. If you are directly creating a PatternEngine instance, then this is not difficult. If you are reusing the entire ConversionDriver, then a bit of extra effort may be there, but I think it is worth it.

2. Instead of hard coding the RUNTIME-QUERY string and explicit getters/setters into the Configuration object, I prefer if you would make the facility generic. That way, future users of this won't have to change Configuration. They should just be able to call addConditionalFlag() or removeConditionalFlag() to configure it.

#32 - 10/22/2013 09:57 AM - Constantin Asofiei

> Code Review 1021c
> 1. It seems like the methods_attributes.rules change should have hwrap as DynamicResource not Dynamicable. But then again, the method name is unwrapDynamicable(). This seems broken.

The handle.unwrap[hwrap] API referenced by the hwrap var in methods_attributes.rules is not hard-linked with the interface name. What it matters is that the returned type defines the APIs which are accessed using the handle.unwrap[hwrap] call. There are no failures related to the name.

> 2. TempTableBuilder._dynamic() is hard coded to true. Why not just return the value of the dynamic data member?

OK, I'll change it.

> Code Review 1021g
> 1. I prefer the enable/disable of conditional rule-set processing to be done at the PatternEngine level.

This will mean that all the rule-sets (even ignored ones) will be parsed. I was thinking to avoid this. If the entire rule container is loaded and the flags are checked when the rule-set is executed, then the flags can not be at the rule-set node in the rule container, they must be in the root rule-set node in the .rules file. And I think this should only prevent rule-set processing (init-/walk-/post-rules) and still allow it to be included (to expose any function libraries).

> That would allow different instances of the pattern engine to be configured differently within the same JVM.

I don't think I understand this one. The flags and the conditional rule-set enable/disable at the Configuration object are context-local, thus flags set by a certain user will be seen only by that context. More, for dyn query purpose, I'm processing each rule-container in its own PatternEngine instance. Do you mean that a PatternEngine holds some JVM-wide data?

> If you are reusing the entire ConversionDriver, then a bit of extra effort may be there, but I think it is worth it.

I didn't find any gain in reusing the ConversionDriver, as for the dyn query purpose, the AST returned by the ProgressParser had to be annotated to specify java-related names (i.e. class, package) and I even had to add bogus ASTs representing a DEFINE TEMP-TABLE tt. statement (with no fields), so that the temp-table name will be converted properly (there is some code in record_scoping_prep.rules which relies on prog.define_temp_table nodes).

> 2. Instead of hard coding the RUNTIME-QUERY string and explicit getters/setters into the Configuration object, I prefer if you would make the facility generic. That way, future users of this won't have to change Configuration. They should just be able to call addConditionalFlag() or removeConditionalFlag() to configure it.

I'm OK with allowing the user to add/remove any flag, but I still want to have a central point to define the constants for the flag names. I think a ConversionFlag enum will be OK.

#33 - 10/22/2013 10:32 AM - Greg Shah

> The handle.unwrap[hwrap] API referenced by the hwrap var in methods_attributes.rules is not hard-linked with the interface name. What it matters is that the returned type defines the APIs which are accessed using the handle.unwrap[hwrap] call. There are no failures related to the name.

OK. But I do prefer it to be consistent (resource name the same as the unwrap method name).

> This will mean that all the rule-sets (even ignored ones) will be parsed. I was thinking to avoid this. If the entire rule container is loaded and the flags are checked when the rule-set is executed, then the flags can not be at the rule-set node in the rule container, they must be in the root rule-set node in the .rules file. And I think this should only prevent rule-set processing (init-/walk-/post-rules) and still allow it to be included (to expose any function libraries).

What I mean here is to move the setConditionalRuleSets()/isConditionalRuleSets() methods and flag into the PatternEngine from the Configuration class. The implementation of conditional processing would still be in ConfigLoader.

> The flags and the conditional rule-set enable/disable at the Configuration object are context-local, thus flags set by a certain user will be seen only by that context.

Yes, this is a good point.

> More, for dyn query purpose, I'm processing each rule-container in its own PatternEngine instance.

Good. And there will be more of this kind of thing in the future (where multiple PE instances will run in the same JVM at the same time) when we implement a fully-multithreaded TRPL.

The key point I am going for here is that this is really best suited as an attribute/flag of the PE, not something that is configured externally.

- We are trying to eliminate/reduce dependence on Configuration.
- This makes the feature more generic which is a useful thing for future TRPL users.

> but I still want to have a central point to define the constants for the flag names. I think a ConversionFlag enum will be OK.

I am OK with an enum, but let's keep it out of code in the pattern/ and cfg/ packages. Those packages are meant to be generic TRPL and should have few (if any) dependencies on P2J or even the conversion driver. Put it in convert/ where we have code that is specific to converting 4GL to Java.

Again, this is consistent with making this a generic improvement to TRPL that future users can take advantage of.

#34 - 10/22/2013 11:14 AM - Constantin Asofiei

Greg, thanks, is much more clear now. Please confirm the following approach is good. At this time, there are two places where a rule-set can be marked:

1. in the rule-container, at the rule-set node (which is a child of a cfg node)
2. in the rule definition .rules file, at the root rule-set node

To me, it looks best to set the flags at the rule-container, and not at the .rules file. I don't want to exclude function libraries included via include. For this, I think is best to use the flags only if a name attribute is given to the rule-set node (i.e. this is a child of the rule-container's root cfg node).

#35 - 10/22/2013 01:19 PM - Greg Shah

I am OK with this approach.

#36 - 10/23/2013 12:21 AM - Eric Faulhaber

The database portions (1021c/g) look OK to me, as long as we address the TODO in TempTableBuilder.asHandle. I have the same question as Greg above: why does the TempTableBuilder._dynamic method always return true instead of the value of dynamic?

#37 - 10/23/2013 07:11 AM - Constantin Asofiei

- File ca_upd20131023a.zip added

- File ca_upd20131023b.zip added

Attached are:

1. 1023a.zip - new version of conditional rule-set processing. In the end, I think is best to allow flags to be set at any non-root rule-set nodes. This way, if a rule-container file contains internal rule-set's, they can be conditionally processed too. I've also added notes to pattern/package.html.
2. 1023b.zip - replacement for 1021c.zip: changes the wrapper from unwrapDynamicable to unwrapDynamicResource and fixes the boolean TempTableBuilder._dynamic(), to return the TempTableBuilder.dynamic flag.

Eric Faulhaber wrote:

> The database portions (1021c/g) look OK to me, as long as we address the TODO in TempTableBuilder.asHandle. I have the same question as Greg above: why does the TempTableBuilder._dynamic method always return true instead of the value of dynamic?

I meant to change it initially, but somehow forgot about it. Is there now.

#38 - 10/23/2013 09:57 AM - Constantin Asofiei

Greg: 1023a.zip needs some more testing, please let me know if the approach is correct.

#39 - 10/23/2013 10:00 AM - Greg Shah

Code Review 1023a/1023b

I am good with all these changes.

The only request I have is to replace the javadoc text in PatternEngine that refers to RUNTIME-QUERY. Please make it generic.

If it passes conversion testing, then do check it in and distribute it.

#40 - 10/23/2013 11:05 AM - Constantin Asofiei

- File ca_upd20131023d.zip added

Greg, please check these changes; they are related to the in-memory AST storage plugin. For this, the filenames must always be normalized. Note that I didn't regression test it yet.

#41 - 10/23/2013 11:36 AM - Greg Shah

Code Review 1023d

1. I really like the changes. It is possible some of the file normalization could cause breakage during conversion processing or possibly during report processing. But if it does cause any problems, it will be because we have some improper processing currently and I would prefer you to fix those locations. The bottom line is that I do believe there may be a few places where we aren't handling the normalization well, but I'd rather have them normalize.

2. The InMemoryRegistryPlugin has this comment: implements an in-memory XML storage option for an AstManagerPlugin. Does it really have anything to do with XML?

3. Should the AstManager.initialize() call (in StandardServer) really be added so early in bootstrap()? It seems to me that unless it is needed very early, it is safer to leave it to later (after registerDefaultServices() seems good). If it doesn't cause a problem to do this, I prefer it because putting it before the directory init and so forth suggests to future readers that there is some important need for this to be done so early. I prefer to make it clear that there is no such need.

#42 - 10/23/2013 11:58 AM - Constantin Asofiei

> 2. The InMemoryRegistryPlugin has this comment: implements an in-memory XML storage option for an AstManagerPlugin. Does it really have anything to do with XML?

No, just ASTs. That should been read "AST storage".

> 3. Should the AstManager.initialize() call (in StandardServer) really be added so early in bootstrap()? It seems to me that unless it is needed very early, it is safer to leave it to later (after registerDefaultServices() seems good). If it doesn't cause a problem to do this, I prefer it because putting it before the directory init and so forth suggests to future readers that there is some important need for this to be done so early. I prefer to make it clear that there is no such need.

I want to initialize the AstManager before anyone else gets a chance to initialize it, as this is a JVM-wide setting. Other classes (like PatterEngine and

SchemaLoader) call AstManager.initialize() in their static c'tor. To me, actually it would look even better if the initialziation is done in the static c'tor for StandardServer.

#43 - 10/23/2013 12:07 PM - Greg Shah

> I want to initialize the AstManager before anyone else gets a chance to initialize it, as this is a JVM-wide setting. Other classes (like PatterEngine and SchemaLoader) call AstManager.initialize() in their static c'tor. To me, actually it would look even better if the initialziation is done in the static c'tor for StandardServer.

I understand the concern, but there shouldn't be any issue with PatternEngine or SchemaLoader executing before the registerDefaultServices() right? Until that point we can't even have converted 4GL code properly run.

If there is ever a time when we need the PatternEngine, SchemaLoader... to be run earlier in the server startup, we can move the AstManager.initialize() accordingly. Until then, let's keep it later rather than sooner.

#44 - 10/23/2013 02:44 PM - Constantin Asofiei

> Code Review 1023d

This has passed conversion testing (as the other 2 updates), but I'll keep it a little more until releasing it, want to make sure the file normalization is stable.

#45 - 10/24/2013 01:48 AM - Constantin Asofiei

- File ca_upd20131023c.zip added

Attached 1023c.zip update is the final version for conditional rule-sets.
The following updates have passed conversion and runtime testing were released:

- 1021g.zip - Annotates the DMO implementation classes with legacy index name mappings. Committed to bzr rev 10404.
- 1023b.zip - Adds conversion and basic runtime support for DYNAMIC attribute. Committed to bzr rev 10405.
- 1023c.zip - Implement conditional rule-set processing. Committed to bzr rev 10406.

#46 - 10/24/2013 10:01 AM - Constantin Asofiei

The plan for implementing the attributes/methods is this:

1. query attributes:
   The following are almost done:

```
NAME
DYNAMIC
ADM-DATA
TYPE
UNIQUE-ID
PRIVATE-DATA
PREPARE-STRING
NUM-BUFFERS
NUM-RESULTS
CURRENT-RESULT-ROW
QUERY-OFF-END
IS-OPEN
FORWARD-ONLY
```

   Following are low priority:

```
INDEX-INFORMATION
```

   Following are not needed for the server project:

```
INSTANTIATING-PROCEDURE
BASIC-LOGGING
CACHE
SKIP-DELETED-RECORD
```

2. query methods:
   QUERY-PREPARE() needs more work/testing, but the following are almost done:

```
GET-CURRENT( )
GET-FIRST( )
GET-LAST( )
GET-NEXT( )
GET-PREV( )
QUERY-CLOSE( )
QUERY-OPEN( )
ADD-BUFFER( )
SET-BUFFERS( )
GET-BUFFER-HANDLE( )
```

   Following are low priority:

```
REPOSITION-FORWARD( )
REPOSITION-TO-ROW( )
REPOSITION-TO-ROWID( )
CREATE-RESULT-LIST-ENTRY( )
DELETE-RESULT-LIST-ENTRY( )
```

   Following are not needed for the server project:

```
DUMP-LOGGING-NOW( )
APPLY-CALLBACK( )
SET-CALLBACK( )
SET-CALLBACK-PROCEDURE( )
GET-CALLBACK-PROC-CONTEXT( )
GET-CALLBACK-PROC-NAME( )
REPOSITION-BACKWARD( )
```

#47 - 10/24/2013 12:03 PM - Constantin Asofiei

Eric: I need the extent information for a field too; should I annotate the inner Composite class or I can assume everything after the "Composite" prefix is the legacy extent?

#48 - 10/24/2013 12:26 PM - Constantin Asofiei

Stanislav: for dynamic temp-tables, I see that you clean all the dynamic DMOs in TempTableBuilder$Context.finished. I probably need something like this for my query builder too, but I don't understand the Context.finished code: why is it dropping all DMOs? Because TEMP-TABLEs can be created in a program executed persistent and AFAIK they need to survive until the persistent program is deleted. But you drop them when the block which called the CREATE TEMP-TABLE statement ends; is there something I'm missing?

#49 - 10/24/2013 12:35 PM - Greg Shah

Do we need a new "hook" to handle the class unloading "event" in a RUN PERSISTENT case? For a procedure run as non-persistent, our Finalizable.finished() which is called when the external procedure ends is the equivalent to closing resources when the procedure is unloaded, since these both happen at the same time. But when a procedure is run persistent, then the external procedure will end (and presumeably Finalizable.finished() will be called) long before the procedure is actually unloaded.

I'm worried this has implications for many resources that use Finalizable.

#50 - 10/24/2013 12:47 PM - Constantin Asofiei

Greg Shah wrote:

> Do we need a new "hook" to handle the class unloading "event" in a RUN PERSISTENT case?

Yes, something like that. In 4GL, the resource might be linked via the INSTANTIATING-PROCEDURE to its "owner": maybe when a persistent procedure is deleted they notify all resources and the ones which have an equal INSTANTIATING-PROCEDURE reference are deleted too? In any case, this is not something query or temp-table specific. I think a separate task is appropriate, to check that all resources (static or dynamic) are deleted properly. And to implement this, I think we will need to register all resources with the current THIS-PROCEDURE instance, at the time of the resource creation; this way, when the procedure gets deleted, the resources can be deleted before the procedure. We might have problems with resources instantiated by the default c'tor, because THIS-PROCEDURE might not be pushed this early (IIRC is pushed when the external procedure block is executed).

#51 - 10/24/2013 12:48 PM - Stanislav Lomany

Constantin Asofiei wrote:

> Stanislav: for dynamic temp-tables, I see that you clean all the dynamic DMOs in TempTableBuilder$Context.finished. I probably need something like this for my query builder too, but I don't understand the Context.finished code: why is it dropping all DMOs? Because TEMP-TABLEs can be created in a program executed persistent and AFAIK they need to survive until the persistent program is deleted. But you drop them when the block which called the CREATE TEMP-TABLE statement ends; is there something I'm missing?

Are you talking about DynamicTablesHelper$Context? If yes, then finished is executed when session ends.

#52 - 10/24/2013 12:49 PM - Eric Faulhaber

Constantin Asofiei wrote:

> Eric: I need the extent information for a field too; should I annotate the inner Composite class or I can assume everything after the "Composite" prefix is the legacy extent?

If you are just looking for the size of the array/extent, you can either get this by parsing the "Composite" class name and taking it from the end, or by invoking the "sizer" methods. Each extent field has a size<PropertyName> method in the DMO, which returns an integer.

#53 - 10/24/2013 12:55 PM - Constantin Asofiei

Stanislav Lomany wrote:

> Constantin Asofiei wrote:
>
> > Stanislav: for dynamic temp-tables, I see that you clean all the dynamic DMOs in TempTableBuilder$Context.finished. I probably need something like this for my query builder too, but I don't understand the Context.finished code: why is it dropping all DMOs? Because TEMP-TABLEs can be created in a program executed persistent and AFAIK they need to survive until the persistent program is deleted. But you drop them when the block which called the CREATE TEMP-TABLE statement ends; is there something I'm missing?
>
> Are you talking about DynamicTablesHelper$Context? If yes, then finished is executed when session ends.

Yes, about that. I didn't notice that you were registering the Finalizable in the global block (which is associated with the user's session). Thanks, now your code makes sense.

#54 - 10/24/2013 12:58 PM - Constantin Asofiei

Eric Faulhaber wrote:

> Each extent field has a size<PropertyName> method in the DMO, which returns an integer.

Thanks, I'll use this.

#55 - 10/24/2013 01:17 PM - Greg Shah

> I think a separate task is appropriate, to check that all resources (static or dynamic) are deleted properly.

Please do create such a task and mark it for milestone 7.

#56 - 10/25/2013 03:14 AM - Constantin Asofiei

Greg Shah wrote:

> > I think a separate task is appropriate, to check that all resources (static or dynamic) are deleted properly.
>
> Please do create such a task and mark it for milestone 7.

See #2196.

#57 - 10/25/2013 09:39 AM - Constantin Asofiei

Eric, I'm finally going into the record scoping behavior for the dynamic queries, and I have a question: why don't we set the RecordBuffer.dmoClass (and RecordBuffer.table too) at the buffer definition, in RecordBuffer.define? This info isn't related to the held record or transaction awareness and IMO access to them should be allowed regardless if the buffer was initialized or not.

#58 - 10/25/2013 10:11 AM - Eric Faulhaber

I didn't want to rely on the naming convention alone to determine the DMO class from the DMO interface. So, in order to determine the DMO class, I needed some other precursor information to set the dmoClass. This information is only realized in the initialize method. If you see a better way to do this, I am not opposed to it in principal.

#59 - 10/25/2013 12:42 PM - Constantin Asofiei

Eric Faulhaber wrote:

> I didn't want to rely on the naming convention alone to determine the DMO class from the DMO interface. So, in order to determine the DMO class, I needed some other precursor information to set the dmoClass. This information is only realized in the initialize method. If you see a better way to do this, I am not opposed to it in principal.

Thanks, after a deeper look is more complex, as it depends on whether the DB is connected or not and other stuff. I'll call initialize() for now before getting the DMO impl class.

On a side note, I think the record scoping for OPEN QUERY is broken in P2J, when a RETURN ERROR is encountered:

```
def var h as handle.

procedure proc0.
   create query h.
   h:add-buffer(buffer book:handle).
   h:query-prepare("for each book").
   h:query-open().
   h:get-first().

   message book.book-id book.book-title.
   return error.
end.

procedure proc0b.
   open query q for each book.
   get first q.
   message book.book-id book.book-title.
   return error.
end.

procedure proc1.
  find first book no-error.
  if not avail book
  then do transaction:
     create book.
     book.book-title = "bla".
     book.isbn = "aaa".
     book.book-id = 10.
  end.
  release book.
end.

/* ensure there is at least one record in the table. */
run proc1.

run proc0 no-error.
if avail(book) then message "Case 1: Book should not be available!".

run proc0b no-error.
if avail(book) then message "Case 2: Book should not be available!".
```

#60 - 10/28/2013 07:40 AM - Constantin Asofiei

I found something interesting and pretty weird about the static QUERY behaviour. Although the query handle CAN be deleted and is deleted, it seems like the query gets "resurrected":

```
def var h1 as handle.
def var h2 as handle.
def temp-table tt1 field f1 as int.

def var i as int.

do i = 1 to 10 transaction:
  create tt1.
  tt1.f1 = i.
end.
release tt1.

def query q for tt1.
open query q for each tt1.
get first q.
h1 = query q:handle.
h2 = query q:handle.

delete object h1.

if valid-handle(h1) <> valid-handle(h2) or h1 <> h2
   then message "handles must be the same!".

h1 = query q:handle. /* here, q points to a different resource !*/

if h1 = h2
   then message "handles must be different!".

if not valid-handle(h1)
   then message "h1 must be valid!".

release tt1.
get next q.
if tt1.f1 <> 2
   then message "incorrect record fetched!"
```

After the delete, both handle vars referring the same resource are both invalid (and can not be used in attr/method calls). But, the static query q can still be used and remains valid! Obtaining the query's handle after it was deleted returns a different resource (which is valid) and somehow remains positioned on the record retrieved by the initial QUERY resource. Can't explain how and why this is happening.

#61 - 10/28/2013 03:34 PM - Eric Faulhaber

I guess this makes sense on some level. For a dynamically created query, the handle(s) to it are the only way to access the backing query. When they are invalidated with the delete object statement, you can no longer reach/use the backing query, and I guess it becomes eligible for resource/garbage collection. However, for a statically defined query, apparently you can still access it with the static reference q after any handle(s) referencing it are invalidated/reassigned.

I have less experience with handles in Progress than you. What behavior would you expect from get next q in the above example? Do you think delete object h1 should have invalidated the static query?

#62 - 10/29/2013 03:31 AM - Constantin Asofiei

Eric Faulhaber wrote:

> I guess this makes sense on some level. For a dynamically created query, the handle(s) to it are the only way to access the backing query. When they are invalidated with the delete object statement, you can no longer reach/use the backing query, and I guess it becomes eligible for resource/garbage collection. However, for a statically defined query, apparently you can still access it with the static reference q after any handle(s) referencing it are invalidated/reassigned.
>
> I have less experience with handles in Progress than you. What behavior would you expect from get next q in the above example? Do you think delete object h1 should have invalidated the static query?

Hmm... this might mean 4GL keeps the static resources and the HANDLE's in distinct structures; because, after DELETE OBJECT is called for a handle referring a query, re-accessing the HANDLE attribute for the static query gives us a different handle ID. This is a little tricky to implement in P2J as, until now, the HANDLE is hard linked with the resource (static or not) and deleting a handle means the resource is deleted too. For now, I'll make the static queries undeletable (regardless how the query is accessed) and I'll document this in a note in #2196.

#63 - 10/29/2013 07:58 AM - Constantin Asofiei

- File ca_upd20131029a.zip added

- File ca_upd20131029c.zip added

1029a.zip is replacement for 1023d.zip (in-memory registry/storage plugin).

1029c.zip contains the changes for conversion of QUERY resources. I didn't include testing-related code in BufferImpl/RecordBuffer/TempTableBuilder which contained mappings for buffer names for testing purposes; the code assumes that the BUFFER:TABLE-HANDLE and BUFFER:NAME attributes are implemented.

Both updates are merged with 10407.

Notes about the 1029c.zip:

- there are failures related to GET NEXT/PREV/etc statements; the behavior in 4GL is the same as with their GET-NEXT/etc method counterparts, but in P2J this does not work properly. I didn't dig deeper into this.
- there are parsing errors related to statements like OPEN QUERY q FOR EACH dbalias.table.. The support in DynamicQueryHelper is added, but the parsing problem is still there.
- Eric: the QUERY-specific APIs in AbstractQuery now throw a RuntimeException, as that implementation should never be reached; the correct implementation class is QueryWrapper (not all of them were moved, the GET methods are still there). I see that in 10407 you had a change

which let some APIS return false, but I don't think this is OK, as it might lead a developer to a wrong path.
- the following files and folders must exist in the server/ folder (i.e. current folder where the server is started):
  - in cfg folder, the matchlist.xml and p2j.cfg.xml files
  - in data/namespace folder, the *.dict and *.p2o files for the standard and all the permanent DBs.

#64 - 10/29/2013 10:35 AM - Greg Shah

Code Review 1029a

This update is good to go. Nice!

Eric will handle the other update.

#65 - 10/30/2013 09:52 AM - Constantin Asofiei

- % Done changed from 0 to 90

- Status changed from WIP to Review

- File ca_upd20131030a.zip added

Attached 1030a.zip (replacement for 1029c.zip) and 1029a.zip have passed conversion and runtime testing. (1029c.zip had a problem related to BROWSE defined with an unopened QUERY).

1029a.zip was committed to bzr revision 10408.

#66 - 10/30/2013 11:22 AM - Constantin Asofiei

I missed something in the QueryWrapper c'tor; I think this code:

```
      // Register a cleaner.  QueryWrappers are created as instance variables,
      // so we register to the next external scope, which corresponds to a
      // business logic class' execute() method.
      Finalizable cleaner = new Finalizable()
      {
         public void finished() { cleanup(); }
         public void iterate() {}
         public void retry() {}
      };
      TransactionManager.registerNextExternal(cleaner);
```

should be executed only for the static query case (as they get defined as instance vars), but need to double-check what cleanup needs to be done for a dynamic query.

#67 - 10/30/2013 12:12 PM - Eric Faulhaber

Constantin Asofiei wrote:

> the following files and folders must exist in the server/ folder (i.e. current folder where the server is started):
> - in cfg folder, the matchlist.xml and p2j.cfg.xml files
> - in data/namespace folder, the *.dict and *.p2o files for the standard and all the permanent DBs.

This is all static data -- either inputs to or outputs from -- conversion. This requirement is OK temporarily for testing, but I think it would be better from a deployment perspective to load this from the application jar file.

#68 - 10/30/2013 12:50 PM - Eric Faulhaber

Code review 20131030a:

Wow, brilliant update! Looks really good, only one major concern: the approach of having a separate, full instance of SchemaDictionary in each context. This is OK for a prototype (especially using a small schema, like p2j_test), but I don't think this is feasible for production use with real world (i.e., potentially huge) schemas, due to the memory requirements of the pristineScope data in each SchemaDictionary instance. We need to think about a more granular breakdown of the class in terms of sharing the pristine data in the global scope and managing the (much smaller) dynamic data per context.

Some other issues, all minor:

- SymbolResolver.locate needs javadoc.
- In newly context-localized classes, when using the same WorkArea instance multiple times in the same method, please cache it and only call locate once. Each call requires a ThreadLocal lookup and multiple hash map lookups in SecurityManager. Not super expensive, but why do it more often than necessary?
- No header entry in UniqueIdGenerator.

Please check it in and distribute, since it already has passed regression testing, but we'll need to figure out some answers for another pass...

#69 - 10/30/2013 01:06 PM - Greg Shah

Let's create new tasks for the open issues:

- Conversion artifacts moved into and read from the application jar.
- Multiple copies of the SchemaDictionary condensed to a single copy with diffs done on a context-local basis.
- Anything else?

These tasks can be worked in M11.

#70 - 10/30/2013 01:31 PM - Constantin Asofiei

Eric Faulhaber wrote:

> ... due to the memory requirements of the pristineScope data in each SchemaDictionary instance.

I think the DynamicQueryHelper.loadPermanentSchemas (which copy the global .dict and .p2o ASTs to each context's AST storage) plus the SchemaDictionar.database map might pose memory problems on the long term (as it copies the same ASTs multiple times).

About the SchemaDictionary.scopes. This is actually a stack, so I'm thinking: why not create a custom stack where the bottom of the stack (scope 0) is shared by all contexts and the other stack levels are context-local? I think this will be OK, as during runtime conversion, scope level 0 should be read-only (no one should add data to it). If it happens, it means that the pristine scope was not initialized properly.

> Some other issues, all minor:

I'll fix these tomorrow (as I need to double-check note 66 too).

> Let's create new tasks for the open issues:

- Anything else?

There are the GET NEXT/PREV/etc problems for both static and dynamic queries; we will need to fix these at some point. I'll put some examples in a dedicated task.

#71 - 10/30/2013 01:37 PM - Eric Faulhaber

Greg Shah wrote:

> Anything else?

The issues Constantin described in notes 63 & 66 above.

With regard to the AbstractQuery issue in note 63, I'm wondering if it makes sense to refactor P2JQuery and AbstractQuery, since so many of their methods can't be used, depending on whether the query is dynamic or not. Don't have a good idea on this yet, but it seems we have outgrown their

initial design...

#72 - 10/30/2013 01:42 PM - Constantin Asofiei

Eric Faulhaber wrote:

> I'm wondering if it makes sense to refactor P2JQuery and AbstractQuery, since so many of their methods can't be used, depending on whether the query is dynamic or not. Don't have a good idea on this yet, but it seems we have outgrown their initial design...

I think we need a QueryResource interface to be implemented only by the QueryWrapper class (in addition of P2jQuery). QueryResource will contain all the QUERY resource APIs (extracted from P2JQuery) and AbstractQuery will have all the APIs defined by the QueryResource removed.

#73 - 10/30/2013 01:46 PM - Eric Faulhaber

Constantin Asofiei wrote:

> > Some other issues, all minor:
>
> I'll fix these tomorrow (as I need to double-check note 66 too).

OK, but tomorrow, please check in the version that already has passed regression testing before you make these changes. I am dependent on your annotations work for some work I am doing, and I would like to avoid waiting for another round of regression testing before merging with your code. Also, I want Stanislav to integrate with your update as soon as possible. The proposed changes shouldn't affect either of us.

#74 - 10/30/2013 01:52 PM - Eric Faulhaber

Constantin Asofiei wrote:

> I think we need a QueryResource interface to be implemented only by the QueryWrapper class (in addition of P2jQuery). QueryResource will contain all the QUERY resource APIs (extracted from P2JQuery) and AbstractQuery will have all the APIs defined by the QueryResource removed.

That sounds good. While we're at it, we should change P2JQuery to Query (or to Queryable, if Query conflicts with Hibernate's version), to address a long-standing complaint of Greg's.

But this refactoring can wait for M11.

#75 - 10/31/2013 02:45 AM - Constantin Asofiei

- File ca_upd20131031a.zip added

Eric Faulhaber wrote:

> OK, but tomorrow, please check in the version that already has passed regression testing before you make these changes.

Attached update was committed to bzr rev 10409. The only changes compared to 1030a.zip are:

- SymbolResolver.locate was added javadoc
- UniqueIdGenerator was added header entry.

#76 - 11/11/2013 02:30 AM - Stanislav Lomany

I've found a minor bug: get-next returns yes even if there are no more results. Testcase:

```
def temp-table tt no-undo field f1 as integer field f2 as integer.
def var q as handle.

create query q.
q:set-buffers(BUFFER tt:HANDLE).
q:query-prepare("for each tt").
q:query-open.
MESSAGE STRING(q:get-next).
```

P2J output: yes, 4GL output: no.
Note that this testcase is not directly converted since legacy temp table names are not supported yet.
I'm not sure about the cause, but it may be about lenientOffEnd.

#77 - 11/14/2013 12:41 PM - Stanislav Lomany

Yet another minor bug. Testcase:

```
def temp-table tt field f1 as integer.

def var h1 as handle.
def var h2 as handle.
def var q as handle.

h1 = buffer book:handle.
h2 = buffer tt:handle.

create query q.
q:set-buffers(h1, h2).
```

```
q:query-prepare("for each book, each tt").
q:query-open.
q:get-next.
```

Warning:

```
[11/14/2013 22:37:35 TMT] (com.goldencode.p2j.persist.Persistence$Context:WARNING) [00000001:00000007:syman-->
local/p2j_test/primary] rolling back open transaction during context cleanup
```

#78 - 11/14/2013 01:06 PM - Stanislav Lomany

Variation of the bug above. Testcase:

```
def temp-table tt0 field f1 as integer.
def temp-table tt field f1 as integer.

def var h1 as handle.
def var h2 as handle.
def var q as handle.

h1 = buffer tt0:handle.
h2 = buffer tt:handle.

create query q.
q:set-buffers(h1, h2).
q:query-prepare("for each tt0, each tt").
q:query-open.
q:get-next.
```

Stacktrace:

```
[11/14/2013 23:04:29 TMT] (com.goldencode.p2j.persist.Persistence$Context:WARNING) [00000001:00000006:syman-->
local/_temp/primary] rolling back open transaction during context cleanup
[11/14/2013 23:04:29 TMT] (com.goldencode.p2j.persist.Persistence$Context:WARNING) [00000001:00000006:syman-->
local/_temp/primary] rolling back orphaned transaction
[11/14/2013 23:04:29 TMT] (com.goldencode.p2j.persist.Persistence$Context:SEVERE) Error closing Hibernate sess
ion during context cleanup
com.goldencode.p2j.persist.PersistenceException: [00000001:00000006:syman-->local/_temp/primary] unable to rol
lback current transaction
        at com.goldencode.p2j.persist.Persistence$Context.rollback(Persistence.java:4615)
        at com.goldencode.p2j.persist.Persistence$Context.closeSession(Persistence.java:5095)
        at com.goldencode.p2j.persist.Persistence$Context.rollback(Persistence.java:4613)
        at com.goldencode.p2j.persist.Persistence$Context.cleanup(Persistence.java:5341)
        at com.goldencode.p2j.persist.Persistence$1.cleanup(Persistence.java:618)
        at com.goldencode.p2j.persist.Persistence$1.cleanup(Persistence.java:616)
        at com.goldencode.p2j.security.ContextLocal$Wrapper.cleanup(ContextLocal.java:381)
        at com.goldencode.p2j.security.SecurityContext.cleanup(SecurityContext.java:498)
        at com.goldencode.p2j.security.SecurityManager.endContext(SecurityManager.java:6772)
        at com.goldencode.p2j.security.SecurityManager.popContextWorker(SecurityManager.java:6710)
        at com.goldencode.p2j.security.SecurityManager.popAndRestoreSecurityContext(SecurityManager.java:3933)
        at com.goldencode.p2j.net.RouterSessionManager.restoreContext(RouterSessionManager.java:1065)
        at com.goldencode.p2j.net.Conversation.run(Conversation.java:191)
        at java.lang.Thread.run(Thread.java:722)
Caused by: org.hibernate.TransactionException: rollback failed
        at org.hibernate.engine.transaction.spi.AbstractTransactionImpl.rollback(AbstractTransactionImpl.java:
```

```
215)
        at com.goldencode.p2j.persist.Persistence$Context.rollback(Persistence.java:4602)
        at com.goldencode.p2j.persist.Persistence$Context.closeSession(Persistence.java:5095)
        at com.goldencode.p2j.persist.Persistence$Context.rollback(Persistence.java:4613)
        at com.goldencode.p2j.persist.Persistence$Context.cleanup(Persistence.java:5341)
        at com.goldencode.p2j.persist.Persistence$1.cleanup(Persistence.java:618)
        at com.goldencode.p2j.persist.Persistence$1.cleanup(Persistence.java:616)
        at com.goldencode.p2j.security.ContextLocal$Wrapper.cleanup(ContextLocal.java:381)
        at com.goldencode.p2j.security.SecurityContext.cleanup(SecurityContext.java:498)
        at com.goldencode.p2j.security.SecurityManager.endContext(SecurityManager.java:6772)
        at com.goldencode.p2j.security.SecurityManager.popContextWorker(SecurityManager.java:6710)
        at com.goldencode.p2j.security.SecurityManager.popAndRestoreSecurityContext(SecurityManager.java:3933)
        at com.goldencode.p2j.net.RouterSessionManager.restoreContext(RouterSessionManager.java:1065)
        at com.goldencode.p2j.net.Conversation.run(Conversation.java:191)
        at java.lang.Thread.run(Thread.java:722)
Caused by: org.hibernate.TransactionException: unable to rollback against JDBC connection
        at org.hibernate.engine.transaction.internal.jdbc.JdbcTransaction.doRollback(JdbcTransaction.java:167)
        at org.hibernate.engine.transaction.spi.AbstractTransactionImpl.rollback(AbstractTransactionImpl.java:
209)
        at com.goldencode.p2j.persist.Persistence$Context.rollback(Persistence.java:4602)
        at com.goldencode.p2j.persist.Persistence$Context.closeSession(Persistence.java:5095)
        at com.goldencode.p2j.persist.Persistence$Context.rollback(Persistence.java:4613)
        at com.goldencode.p2j.persist.Persistence$Context.cleanup(Persistence.java:5341)
        at com.goldencode.p2j.persist.Persistence$1.cleanup(Persistence.java:618)
        at com.goldencode.p2j.persist.Persistence$1.cleanup(Persistence.java:616)
        at com.goldencode.p2j.security.ContextLocal$Wrapper.cleanup(ContextLocal.java:381)
        at com.goldencode.p2j.security.SecurityContext.cleanup(SecurityContext.java:498)
        at com.goldencode.p2j.security.SecurityManager.endContext(SecurityManager.java:6772)
        at com.goldencode.p2j.security.SecurityManager.popContextWorker(SecurityManager.java:6710)
        at com.goldencode.p2j.security.SecurityManager.popAndRestoreSecurityContext(SecurityManager.java:3933)
        at com.goldencode.p2j.net.RouterSessionManager.restoreContext(RouterSessionManager.java:1065)
        at com.goldencode.p2j.net.Conversation.run(Conversation.java:191)
        at java.lang.Thread.run(Thread.java:722)
Caused by: org.h2.jdbc.JdbcSQLException: The object is already closed [90007-169]
        at org.h2.message.DbException.getJdbcSQLException(DbException.java:329)
        at org.h2.message.DbException.get(DbException.java:169)
        at org.h2.message.DbException.get(DbException.java:146)
        at org.h2.message.DbException.get(DbException.java:135)
        at org.h2.jdbc.JdbcConnection.checkClosed(JdbcConnection.java:1386)
        at org.h2.jdbc.JdbcConnection.checkClosedForWrite(JdbcConnection.java:1374)
        at org.h2.jdbc.JdbcConnection.rollback(JdbcConnection.java:460)
        at org.hibernate.engine.transaction.internal.jdbc.JdbcTransaction.doRollback(JdbcTransaction.java:163)
        at org.hibernate.engine.transaction.spi.AbstractTransactionImpl.rollback(AbstractTransactionImpl.java:
209)
        at com.goldencode.p2j.persist.Persistence$Context.rollback(Persistence.java:4602)
        at com.goldencode.p2j.persist.Persistence$Context.closeSession(Persistence.java:5095)
        at com.goldencode.p2j.persist.Persistence$Context.rollback(Persistence.java:4613)
        at com.goldencode.p2j.persist.Persistence$Context.cleanup(Persistence.java:5341)
        at com.goldencode.p2j.persist.Persistence$1.cleanup(Persistence.java:618)
        at com.goldencode.p2j.persist.Persistence$1.cleanup(Persistence.java:616)
        at com.goldencode.p2j.security.ContextLocal$Wrapper.cleanup(ContextLocal.java:381)
        at com.goldencode.p2j.security.SecurityContext.cleanup(SecurityContext.java:498)
        at com.goldencode.p2j.security.SecurityManager.endContext(SecurityManager.java:6772)
        at com.goldencode.p2j.security.SecurityManager.popContextWorker(SecurityManager.java:6710)
        at com.goldencode.p2j.security.SecurityManager.popAndRestoreSecurityContext(SecurityManager.java:3933)
        at com.goldencode.p2j.net.RouterSessionManager.restoreContext(RouterSessionManager.java:1065)
        at com.goldencode.p2j.net.Conversation.run(Conversation.java:191)
        at java.lang.Thread.run(Thread.java:722)
```

#79 - 12/05/2013 07:24 PM - Eric Faulhaber

- Subject changed from prototype simple runtime support for dynamically prepared queries to add runtime support for dynamically prepared queries

#80 - 01/07/2014 12:29 PM - Eric Faulhaber

- Status changed from Review to Closed

- % Done changed from 90 to 100

The problematic test cases above have been recorded in #2220, as they are not expected to be needed for M7.

#81 - 11/16/2016 11:42 AM - Greg Shah

- Target version changed from Milestone 7 to Runtime Support for Server Features

Files

ca_upd20131021c.zip	78.3 KB	10/21/2013	Constantin Asofiei
delete_static_resource.p	1.9 KB	10/21/2013	Constantin Asofiei
ca_upd20131021d.zip	31.4 KB	10/21/2013	Constantin Asofiei
ca_upd20131021g.zip	31 KB	10/21/2013	Constantin Asofiei
ca_upd20131021f.zip	18.4 KB	10/21/2013	Constantin Asofiei
ca_upd20131023a.zip	68.1 KB	10/23/2013	Constantin Asofiei
ca_upd20131023b.zip	78.3 KB	10/23/2013	Constantin Asofiei
ca_upd20131023d.zip	66.3 KB	10/23/2013	Constantin Asofiei
ca_upd20131023c.zip	68.1 KB	10/24/2013	Constantin Asofiei
ca_upd20131029a.zip	66.5 KB	10/29/2013	Constantin Asofiei
ca_upd20131029c.zip	340 KB	10/29/2013	Constantin Asofiei
ca_upd20131030a.zip	340 KB	10/30/2013	Constantin Asofiei
ca_upd20131031a.zip	340 KB	10/31/2013	Constantin Asofiei