# Database - Feature #1910

Feature # 1582 (Closed): add conversion and runtime support for sequences

## add schema conversion support for sequences

11/05/2012 11:23 PM - Eric Faulhaber

| | | | |
|---|---|---|---|
| **Status:** | Closed | **Start date:** | 11/12/2012 |
| **Priority:** | Normal | **Due date:** | |
| **Assignee:** | Ovidiu Maxiniuc | **% Done:** | 100% |
| **Category:** | | **Estimated time:** | 126.00 hours |
| **Target version:** | Conversion Support for Server Features | | |
| **billable:** | No | **version:** | |
| **vendor_id:** | GCD | | |

**Description**

**Related issues:**

| | |
|---|---|
| Blocks Database - Feature #1911: add runtime support for sequences | **Closed** |

### History

**#1 - 11/05/2012 11:28 PM - Eric Faulhaber**

*- Status changed from New to Hold*

**#2 - 11/05/2012 11:28 PM - Eric Faulhaber**

*- Target version set to Milestone 3*

**#3 - 11/06/2012 09:37 AM - Eric Faulhaber**

*- Start date deleted (11/05/2012)*

**#4 - 11/12/2012 09:21 AM - Eric Faulhaber**

*- Start date set to 11/12/2012*

*- Status changed from Hold to New*

See parent issue for description of what must be converted.

The statements in the schema export (.df) file must be converted to DDL, which may be dialect-specific by database vendor. I can't think of an existing class to which it would be appropriate to add the converted language statement (CURRENT-VALUE) and built-in functions (NEXT-VALUE and CURRENT-VALUE) as methods, so maybe we need a new Sequence class. Initially just stub out the methods; there is a separate issue (#1911) for the runtime implementation.

**#5 - 11/16/2012 11:35 AM - Ovidiu Maxiniuc**

*- Status changed from New to WIP*

**#6 - 11/16/2012 11:57 AM - Greg Shah**

Please show the changes you have had to make. I am especially interested in the progress.g changes, but show everything.

**#7 - 11/19/2012 02:04 AM - Ovidiu Maxiniuc**

I created the new com.goldencode.p2j.util.Sequence class with the appropriate methods declarations.
However, the Progress antlr generator failed to create a parser that handle correctly the OE syntax so progress.g suffered the following changes:

```
current_value_stmt:
      KW_CUR_VAL^ LPARENS! sequence_ref (COMMA! database_name[false])? RPARENS! EQUALS! expr;

dynamic_current_value_stmt:
      KW_DYN_CURV^ LPARENS! expr COMMA! expr RPARENS! EQUALS! expr;

sequence_funcs:
      (KW_CUR_VAL^ | KW_NEXT_VAL^) LPARENS! sequence_ref (COMMA! database_name[false])? RPARENS!
      {
         // save the original token type, then rewrite the type
         ##.putAnnotation("oldtype", new Long(##.getType()));
         ##.setType(FUNC_INT);

         // write our function-specific annotations
         sym.annotateFunction(##.getText(), ##, false);
      };

dynamic_sequence_funcs:
      (KW_DYN_CURV^ | KW_DYN_NEXV^) LPARENS! expr COMMA! expr RPARENS!
      {
         // save the original token type, then rewrite the type
         ##.putAnnotation("oldtype", new Long(##.getType()));
         ##.setType(FUNC_INT64);

         // write our function-specific annotations
         sym.annotateFunction(##.getText(), ##, false);
      };

primary_expr:
      {
      ...
      (
         options { generateAmbigWarnings = false; }
         :
           literal

           // must be before record_funcs
         | { LA(2) != LPARENS && !sym.isTable(LT(2).getText()) }?
           new_phrase

           // these next functions have specialized signature processing, but
           // they otherwise work normally (from a syntax perspective), so
           // they must occur before func_call since it would otherwise
           // conflict
         | { embeddedSql }?
           sql_aggregate_funcs
         | seek_func
         | cast_func
         | sequence_funcs
         | dynamic_sequence_funcs
         | dynamic_function_func
         | frame_funcs
         | record_funcs
    ...
```

These changes are needed for 2 main causes:

- dynamic forms receive as parameters always two character expressions instead of identifiers (ldbName not optional)
- the sequence statements were declared as functions, not as statements

#### #8 - 11/19/2012 09:27 AM - Greg Shah

Questions:

```
the sequence statements were declared as functions, not as statements
```

The changes to "current_value_stmt" and "dynamic_current_value_stmt" are not supposed to have the "EQUALS! expr" at the end. Those rules are called from "assign_type_syntax_stmt_list", which is called from "lvalue", which is matched as the left operand of the assignment operator in the "assignment" rule. The "assignment" rule is where we attempt to differentiate between the statement (which looks like a function as an lvalue taking an assignment) and real function calls. In particular, we attempt to differentiate between standalone expressions and assignments. We use the normal assignment processing to also handle the "assign style statements" of which there are more than a few. It is poorly designed 4GL syntax for sure.

Do you have specific code examples where the "current_value_stmt" and "dynamic_current_value_stmt" were not parsed properly? If so, please post those here. We would solve that problem differently.

```
dynamic forms receive as parameters always two character expressions instead of identifiers (ldbName not optio
nal)
```

This would be a good thing. But I want to be clear: have you tested trying to embed an unquoted logical database name instead of a character type (or literal) in the 2nd argument of the dynamic sequence functions?

If it really always is a character type, then instead of creating a new "dynamic_sequence_funcs" rule, we can do this even simpler. Just add "dynamic-current-value" and dynamic-next-value as FUNC_INT64 in the initializeFunctionDictionary() array. All functions that take a list of postfixed and parenthesized expressions as parameters can be matched there. The only reason we have the "sequence_funcs" rule (and its custom entry in "primary_expr") is to allow the use of an unquoted logical DB name (which is not something we otherwise match as an expression).

#### #9 - 11/19/2012 09:48 AM - Ovidiu Maxiniuc

The answer for 2nd question for now:
I am positive that DYNAMIC- forms require exactly 2 character parameters. This is what the .pdf doc affirms. I also tested this so:

- if I use use an identifier (of a sequence for 1st param or database for 2nd) instead of char expression I get error 201: Unknown Field or Variable name.
- if I try to use only one parameter, I get error 12377: DYNAMIC-NEXT-VALUE and DYNAMIC-CURRENT-VALUE require the second DBNAME argument.

I believe this is the difference, the DYNAMIC parameters can be expressions, evaluated dynamically at runtime, while the simple forms need statically declared identifiers for both sequence and database. The return type switch from integer to int64 seems like a normal upgrade as the dynamic forms are newer.

BTW, what is the p2j 64bit-wide numeric type ?

#### #10 - 11/19/2012 09:54 AM - Greg Shah

We have not yet written the backing class for the 64-bit integer. It will be called int64 when it is written. But the parser already supports the various forms, including FUNC_INT64 for a function that returns an INT64 type.

Based on your results, go ahead with putting the "dynamic-current-value" and "dynamic-next-value" into the initializeFunctionDictionary() array.

#### #11 - 11/19/2012 10:02 AM - Ovidiu Maxiniuc

As far as I can see, they are already there:

```
sym.addBuiltinFunction("dynamic-current-value", FUNC_INT64      , false);
...
sym.addBuiltinFunction("dynamic-next-value"   , FUNC_INT64      , false);
```

#### #12 - 11/19/2012 10:14 AM - Greg Shah

OK, that explains things. When I added support for those, I had no way to test if the unquoted names could be used in addition to the character expression forms. So I must have added support in both places. The solution is just to remove the extra support that used to be in the "sequence_funcs" rule.

#### #13 - 11/19/2012 10:53 AM - Ovidiu Maxiniuc

What do you mean?
At this moment, the P2J work just fine, .p file is correctly converted.
Keeping sequence_funcs is important as it forces P2J to use identifiers in static forms of the functions.
Here is the error I got when I convert seq-val-2 = CURRENT-VALUE(my-seq-1, p2j_test).:

```
line 65:27: unexpected token: my-seq-1
```

On the other hand, the dynamic forms indeed do not need such thing as they are normal functions that can have any expression as parameters as long as they evaluate to the name of sequence/database.

#### #14 - 11/19/2012 10:54 AM - Ovidiu Maxiniuc

I can now confirm that the statements also need "EQUALS! expr" at the end. Indeed, the syntax is stupid, but the antlr handles it well, P2J will convert the statement: CURRENT-VALUE(my-seq-1) = 10 + 5. into:

```
Sequence.currentValue("my-seq-1", plus(10, 5)));
```

Droping the "EQUALS! expr" at the end, the same statement will be converted to :

```
"my-seq-1".assign(plus(10, 5));
```

I wold prefer it this way also because it reflects exactly the syntax from the .pdf manuals.

#### #15 - 11/19/2012 11:40 AM - Greg Shah

```
On the other hand, the dynamic forms indeed do not need such thing as they are normal functions that can have
any expression as parameters as long as they evaluate to the name of sequence/database.
```

Exactly.  These need to be removed from sequence_funcs.

```
I can now confirm that the statements also need "EQUALS! expr" at the end.
```

OK, show the 4GL that breaks our code.  We are definitely NOT going to add the "EQUALS! expr" at the end.  Instead we are going to fix whatever problem there is with not matching properly in the assignment rule.

#### #16 - 11/19/2012 11:43 AM - Greg Shah

The key point here is that the assignment style statements should already be working without duplicating the assignment syntax in many places in the code.  So if you have code that breaks it, we need to understand why that is happening.

#### #17 - 11/19/2012 01:33 PM - Greg Shah

From progress.g:

```
...
 * This rule is consolidated to allow these constructs to be accessed from
 * {@link #lvalue} which allows then to be in an {@link #assign_stmt} which
 * actually calls this rule via {@link #assign}.  Top level use of these as
 * a language statement is handled via the {@link #assignment} rule which
 * also handles the matching for the EQUALS and following
 * {@link #expr}.
 */
assign_type_syntax_stmt_list
```

You are right that directly encoding the "EQUALS! expr" at the end is easier to read and is preferable.  As a standalone "expression", it could be matched as a language statement:

dynamic-current-value("my-sequence", "my-database") = 1000.

But unfortunately, in Progress 4GL, they also allow you to do this:

```
ASSIGN my-var = 5
dynamic-current-value("my-sequence", "my-database") = 1000
another-var = "whatever".
```

In other words, this is treated more generally than just a normal language statement, it is treated like an lvalue. That is why we match it in such a backwards manner.

#### #18 - 11/19/2012 05:39 PM - Eric Faulhaber

So far the discussion has centered around the conversion of business logic references to sequences, but the highest priority portion of this issue must be:

1. the determination of whether we can use native database sequences to support this feature set, and if so...
2. the generation of DDL to create/configure those native sequences.

This functionality is required for Milestone 3 and blocks #1917. Please focus on it first.

#### #19 - 12/06/2012 02:35 AM - Ovidiu Maxiniuc

I observed yesterday that the converted data dictionary file .p2o does not contains any reference to the sequence. The .dict and .schema files seem generated fine. This means that in spite of the generation of DDL that create the sequences are ok, I believe that the current value cannot be imported.
I must note that the current values of all sequences of a database are exported in a separate .d file even though the other properties (init, max, cycle, .. values) are exported into the same .df as the tables.
I suppose I have to add support for setting the current values of sequences at the same time with table - data import but from a dedicated .d file.

#### #20 - 12/06/2012 09:53 AM - Eric Faulhaber

I'm sorry, but in my review of your initial updates, I didn't notice that you were generating the sequence DDL from the .schema file instead of from the .p2o file. It really should be the latter. This means that the sequence information should be pushed into the .p2o file the way we do with all the other schema information. This is driven from p2o.xml (but it probably should go in a new rule set that is called from there, like we do with p2o_composite.rules). As you note, this also will allow us to drive the sequence current value import at the same time as the table data import.

How much effort is it to rework the sequence DDL generation in this way?

#### #21 - 12/06/2012 10:23 AM - Ovidiu Maxiniuc

Taking into consideration that the rules are more or less the same, a couple of hours should be enough to move the source of sequence DDLs.
But before that the .p2o should be aware of the sequence objects.

#### #22 - 12/06/2012 10:28 AM - Eric Faulhaber

Please make this issue your highest priority (including the import of current sequence values). We need to deliver a converted/imported test database very soon. The runtime support is less urgent, so please split your update and provide one that just includes the sequence conversion support for review, as soon as this feature set is ready.

#### #23 - 12/07/2012 02:01 PM - Eric Faulhaber

*- % Done changed from 0 to 40*

#### #24 - 12/10/2012 06:59 AM - Ovidiu Maxiniuc

*- File om_upd20121210a.zip added*

Moved the generation of sequence ddl from .schema to .p2o file.

#### #25 - 12/10/2012 11:11 AM - Ovidiu Maxiniuc

*- File om_upd20121210b.zip added*

*- Status changed from WIP to Review*

Added sequence current value import

#### #26 - 12/10/2012 01:38 PM - Eric Faulhaber

Code review 1210a:

Please note that the following files in your zip file:

```
src/com/goldencode/schema/
src/com/goldencode/schema/DataModelTokenTypes.java
src/com/goldencode/schema/DataModelAst.java
src/com/goldencode/uast/
src/com/goldencode/uast/progress.g
src/com/goldencode/persist/
src/com/goldencode/persist/dialect/
src/com/goldencode/persist/dialect/P2JDialect.java
src/com/goldencode/persist/dialect/P2JH2Dialect.java
src/com/goldencode/persist/dialect/P2JPostgreSQLDialect.java
src/com/goldencode/persist/SequenceManager.java
```

are missing a p2j subdirectory under goldencode, so it would make a bit of a mess if you were to unzip it over your working directory.

Other than that, it looks quite good overall.  Some relatively minor points:

- generate_seq_ddl.rules includes schema/import_util.  Is it used?
- builtin_functions.rules:  the dbimport variable can be re-used for flagging the need to add the com.goldencode.p2j.persist.* import statement, rather than creating the seqimp variable for the same purpose.  Was there a specific reason you added the new variable?
- DataModelAst.java:  comment in header references "p2j" file, but I think you mean "p2o" file.
- P2JH2Dialect.java, P2JPostgreSQLDialect.java (minor formatting issue):  please refer to previous code review comments (#1582, note 14) regarding aligning chained method calls across lines.

#### #27 - 12/10/2012 01:38 PM - Eric Faulhaber

*- % Done changed from 40 to 60*

*- Status changed from Review to WIP*

#### #28 - 12/10/2012 02:12 PM - Eric Faulhaber

Code review 1210b:

- Nice work!  I'm excited to give this a test run with a real database.
- Is _seqvals.d a hard-coded/well-known name that Progress uses when exporting sequence current values? Since we are hard-coding it into the import ruleset, this fact should be well documented.
- import.xml:  Please remove any temporarily added debug statements (like at lines 447 and 725).
- ImportWorker.java:  other than a SQLException when closing a Statement or Connection, please log more detail on any abnormal result.  For example, there are places where we detect a malformed line or a sequence name that was not added to the collection by the ruleset.  In both cases, we continue without any warning being written, so the user is unaware of these abnormalities.
- Minor formatting issues:  please ensure all indents are 3 spaces (see last finally statement in initializeSequences method), and that javadoc formatting is consistent with other methods.

#### #29 - 12/10/2012 02:19 PM - Eric Faulhaber

After you finish with the code changes resulting from the above code reviews, please review all places in the P2J Conversion Handbook and P2J Conversion Reference where the topics of schema conversion, DDL generation, and data import are discussed, and update these accordingly to account for the sequence-related features you have just added.

#### #30 - 12/11/2012 11:29 AM - Ovidiu Maxiniuc

*- File om_upd20121211a.zip added*

Changes in today's update:

- src directory structure fixed
- dbimport replaced seqimp
- fixed indentation/alignment, comments and javadocs, and debug statement removed
- errors (warnings - I would say) printed into console when importing/setting the current value of sequences, also printed the list of sequences that were NOT initialized because they were not found in export file.
- fixed a couple of bugs I have discovered

The schema/import_util cannot be removed from generate_seq_ddl.rules as it is needed indirectly from schema/dmo_common.rules, hiberProps use buildHibernateProperties function for initialization.

The _seqvals.d is indeed the default filename for exported sequence values. I documented this in the xml script file along with document/page references. However, as the user that performs the export is free to change the default filename I think we should offer some mechanism to read the values from the file (s)he exported. Is there a way to read this filename form a configuration file ? or should I add an optional command line parameter for PatternEngine.main() ?

#### #31 - 12/11/2012 12:05 PM - Eric Faulhaber

Ovidiu Maxiniuc wrote:

> The _seqvals.d is indeed the default filename for exported sequence values. I documented this in the xml script file along with document/page references. However, as the user that performs the export is free to change the default filename I think we should offer some mechanism to read the values from the file (s)he exported. Is there a way to read this filename form a configuration file ? or should I add an optional command line parameter for PatternEngine.main() ?

We could make this a configuration option in p2j.cfg.xml, but I don't see much value in doing this. Only one database is imported at a time and the dump files are stored together in a single directory, so there's no need to differentiate the name of this file for different databases. Let's just document that using the default name is a requirement for now. If we see value in making this configurable later, we can always do so.

#### #32 - 12/14/2012 12:15 PM - Ovidiu Maxiniuc

*- File om_upd20121214a.zip added*

As I created a new test case file for sequences I discovered that the Sequencemanager.setValue() set of overloaded methods does not cover all cases, because the converter can choose to use directly java strings for sequence name and database if those expressions are so simple.

Also, I got a strange error today when converting the application.

```
EXPRESSION EXECUTION ERROR:
---------------------------
bdt = create(cls)
      ^  { com.goldencode.p2j.util.int64 }
---------------------------
EXPRESSION EXECUTION ERROR:
---------------------------
fmt = execLib("format_for_type", ref)
      ^  { Expression execution error @1:7 }
---------------------------
EXPRESSION EXECUTION ERROR:
---------------------------
fmtst1 = execLib("get_override_format_string", copy)
         ^  { Expression execution error @1:7 }
---------------------------
ERROR:
java.lang.RuntimeException: ERROR!  Active Rule:
-----------------------
      RULE REPORT
-----------------------
Rule Type :   WALK
Source AST:  [ expression ] BLOCK/STATEMENT/KW_DISP/EXPRESSION/ @0:0 {12884902068}
Copy AST  :  [ expression ] BLOCK/STATEMENT/KW_DISP/EXPRESSION/ @0:0 {12884902068}
Condition :  bdt = create(cls)
Loop      :  false
--- END RULE REPORT ---
```

The only (temporary) solution I found is to declare the dynamic forms of dynamic-next-value and dynamic-current-value to have a FUNC_INT type instead of FUNC_INT64 in progress.g. I believe the int64 is in development and I must have merged something from repository.
I added a new update with these two changes even if the later is a temporary fix.

#### #33 - 12/20/2012 06:28 PM - Eric Faulhaber

Code review 1214a:

Is the version of SequenceManager.java in this update the correct one? It doesn't compile. First, there was no closing curly brace for the class. After I fixed this, I got the following errors upon compilation:

```
compile:
    [javac] Compiling 993 source files to /home/ecf/projects/p2j/oldrpt/build/classes
    [javac] /home/ecf/projects/p2j/oldrpt/src/com/goldencode/p2j/persist/SequenceManager.java:23: error: packa
ge com.goldencode.p2j.persist.sequence does not exist
    [javac] import com.goldencode.p2j.persist.sequence.SequenceHandler;
    [javac]                                                 ^
    [javac] /home/ecf/projects/p2j/oldrpt/src/com/goldencode/p2j/persist/dialect/P2JH2Dialect.java:545: error:
 cannot find symbol
    [javac]          "CREATE CACHED TABLE " + SequenceManager.SEQ_TABLE + " (" +
    [javac]                                                   ^
    [javac]   symbol:   variable SEQ_TABLE
    [javac]   location: class SequenceManager
    [javac] /home/ecf/projects/p2j/oldrpt/src/com/goldencode/p2j/persist/dialect/P2JH2Dialect.java:586: error:
 cannot find symbol
    [javac]           "INSERT INTO " + SequenceManager.SEQ_TABLE +
    [javac]                                                 ^
    [javac]   symbol:   variable SEQ_TABLE
    [javac]   location: class SequenceManager
    [javac] /home/ecf/projects/p2j/oldrpt/src/com/goldencode/p2j/persist/dialect/P2JPostgreSQLDialect.java:725
: error: cannot find symbol
    [javac]          "CREATE TABLE \"" + SequenceManager.SEQ_TABLE + "\" (" +
    [javac]                                                 ^
    [javac]   symbol:   variable SEQ_TABLE
    [javac]   location: class SequenceManager
    [javac] /home/ecf/projects/p2j/oldrpt/src/com/goldencode/p2j/persist/dialect/P2JPostgreSQLDialect.java:735
: error: cannot find symbol
    [javac]          "   CONSTRAINT \"" + SequenceManager.SEQ_TABLE + "_pkey\"" +
    [javac]                                                  ^
    [javac]   symbol:   variable SEQ_TABLE
    [javac]   location: class SequenceManager
    [javac] /home/ecf/projects/p2j/oldrpt/src/com/goldencode/p2j/persist/dialect/P2JPostgreSQLDialect.java:768
: error: cannot find symbol
    [javac]           "INSERT INTO \"" + SequenceManager.SEQ_TABLE + "\"" +
    [javac]                                                 ^
    [javac]   symbol:   variable SEQ_TABLE
    [javac]   location: class SequenceManager
    [javac] Note: Some input files use unchecked or unsafe operations.
    [javac] Note: Recompile with -Xlint:unchecked for details.
    [javac] 6 errors
```

I needed to compile and use the sequence support, so I removed the import statement in SequenceManager and replaced the implementations of the getP2JCreateSequenceTable and the getP2JAddSequenceString methods, such that they just return empty string for now, as I could not find any callers of these methods. Are they used, or are they left over from an earlier design? If not in use, please remove them.

#### #34 - 12/21/2012 12:35 PM - Eric Faulhaber

I have merged the 1214a update with my local changes and tested a real-world database import with sequences.  I found some additional issues related to sequence conversion, which I have fixed (but I still need to regression test and check them in).  Specifically:

- sequence names need to be converted to SQL-compliant names using the NameConversionWorker;
- there was a latent defect in DataModelWorker.Library.sortClasses, which would drop all sequences for a schema with more than one table.

#### #35 - 12/21/2012 12:37 PM - Eric Faulhaber

*- % Done changed from 60 to 70*

#### #36 - 12/28/2012 11:58 AM - Ovidiu Maxiniuc

This note is somewhere at the border between sequence conversion and runtime.
I tried to add the H2 database support for sequences the way Greg mentioned in a previous note: using the directory.
I wrote most of the code, including:

- adding a new getter in P2JDialect that flags if the current database dialect needs extra support (this is true for H2, Postgres (and the future MSSQL) servers fully support 4GL features)
- adding code in import.xml and ImportWorker.java that read sequence data from .p2o and attempt to store them to directory.xml for future runtime processing
- adding Long/int64 nodes accessors in DirectoryService (including remappers) as sequences use only int64 values
- adding P2J management for bounded/cycle sequences

I have some issues / questions about all these modification:
1. I did not have access to directory during the import phase. I am not fully aware if this is even possible. I see that the import process is rather sandboxed, it only have access to datafiles (.p2o and .d) and database. I understand that I can force the creation of a DirectoryService by using a bootstrap configuration identical with the one of server runtime, but I'm afraid this is not the right thing to do.
2. I could not see the effects of my new Long/int64 node implementation of DirectoryService/remappers also because of 1. and they are written blindly, only with the help of the compiler.

#### #37 - 01/02/2013 01:10 AM - Eric Faulhaber

*- % Done changed from 70 to 90*

Sequence conversion and import for PostgreSQL seems to work well with some minor changes I've made.  I need to merge these with main trunk and check into bzr.

#### #38 - 01/02/2013 07:23 PM - Eric Faulhaber

Ovidiu Maxiniuc wrote:

> ...
> I have some issues / questions about all these modification:
> 1. I did not have access to directory during the import phase. I am not fully aware if this is even possible. I see that the import process is rather sandboxed, it only have access to datafiles (.p2o and .d) and database. I understand that I can force the creation of a DirectoryService by using a bootstrap configuration identical with the one of server runtime, but I'm afraid this is not the right thing to do.

You are correct, the directory is a run-time concept, not conversion-time. Since you need to take information from the _seqvals.d file at import that will be stored in the directory, I think we will have to generate an XML "snippet" that can be merged into the directory later. Greg does something similar to this with the mapping of Progress procedures to Java classes. If you search for the string "name_map.xml" throughout the P2J source (in *.java, *.xml, and *.rules files), you will see how he does this.

#### #39 - 01/04/2013 09:36 AM - Greg Shah

I thought that the "directory-backed crossed-the-sequence-boundary flag" approach was only needed to remember the fact that the last next-value call crossed the boundary and that the current-value must return ? instead of the last valid value. But in a conversion situation, we have a one-time migration going on. There is no real knowledge of whether we have crossed a boundary or not, is there?

In other words, is there really something to migrate here, or can we assume that all the flags are always false and they can be persisted by the server runtime to the correct values the first time the server shuts down?

#### #40 - 01/04/2013 10:15 AM - Ovidiu Maxiniuc

When doing the (one-time) import there is no special care for sequences, they are imported with the actual value. However, because int the H2 there is no notions of "bounded" and cycle sequences, the max and min values are lost if not saved *somewhere* and taken into consideration afterwards.

Let's take an example: Seq-1 (Start:0, Inc:1, Min:0, Max:5, Cycle:No, CrtVal:4).
When doing the import, the only kept properties are: Start, Inc and CrtVal. The other cannot be set by H2 sequence statements so, if they are lost, the next two generated values will be 5 and 6 instead of 5 and ?. (As a parenthesis here, if Seq-1 would have the Cycle attribute set then the next two values are 5 and 0.) This is not the way 4GL works so it's wrong.
If at runtime P2J has the missing attributes it can check them act act accordingly, returning ? if upper/lower bound is crossed or warp around back to min value (and set it also to H2) in the case of cycle sequences.

On the other hand, (based on the recent discoveries that the documentation is wrong about current-value) the unknown value (?) will never be exported, as current-value will return the last valid value of the sequence and no special case need to be treated at import time.

The way I see the fix for this issue the h2 unsupported sequence attributes must be saved and later queried from a persistent location. As using a dedicated table in database interferes badly with other transactions and the workaround code is cumbersome and a performance penalty, the solution is the directory or other kind of file where these values are persisted between server restarts (assuming that at runtime the values are stored/managed in process' memory).
Am I wrong? Is there another solution I fail to notice ?

#### #41 - 01/04/2013 10:40 AM - Greg Shah

OK, I understand that there is some configuration that needs to be migrated, when the backing database does not support the feature.

For PostgreSQL, there is nothing extra to migrate?

For H2, there is the min, max and cycle values?

If so, then there isn't any variable data that is being migrated. Rather, it is sequence configuration information. Perhaps it is better to put that into a more static schema-specific file such as the dmo-index.xml?

**#42 - 01/04/2013 12:26 PM - Ovidiu Maxiniuc**

> OK, I understand that there is some configuration that needs to be migrated, when the backing database does not support the feature.

Yes, there are some essential information that is needed later, when calling next-value() function.

> For PostgreSQL, there is nothing extra to migrate?

No, PostgreSQL has support for all 4GL features.

> For H2, there is the min, max and cycle values?

Yes. H2 only support non-terminating sequences. (At least not until 9223372036854775807 (Long.MAX_VALUE) when it will silently warp around to Long.MIN_VALUE). In fact, they are some kind of cycled sequences over the Long range of integers.

> If so, then there isn't any variable data that is being migrated. Rather, it is sequence configuration information. Perhaps it is better to put that into a more static schema-specific file such as the dmo-index.xml?

Probably. I understand that sequences cannot be managed from 4GL programming language, only from the Data Dictionary tool and I believe we can assume they normally will rarely change (if ever). They are additional static schema information.

Does-it make sense to create another file or should I just extend the dmo-index's dtd schema element with another 'sequence' element with the attributes we need ?
I don't know how the parsing is done at runtime, doing the 2nd way could be a breaker if the parser assumes that there are only class -es nodes into a schema.

#### #43 - 01/04/2013 02:07 PM - Eric Faulhaber

Ovidiu Maxiniuc wrote:

> Does-it make sense to create another file or should I just extend the dmo-index's dtd schema element with another 'sequence' element with the attributes we need ?

The latter option: extend dmo-index.

> I don't know how the parsing is done at runtime, doing the 2nd way could be a breaker if the parser assumes that there are only class -es nodes into a schema.

See the DMOIndex class in the persist package (the updateMap method, particularly). Currently, there is no such assumption; it walks the child nodes based on their tag name.

#### #44 - 01/09/2013 09:05 AM - Ovidiu Maxiniuc

*- File om_upd20130109a.zip added*

*- Status changed from WIP to Review*

The sequences are now imported directly into psql using generated ddl from conversion step and initialized at the import stage.
For H2 they are specially added to dmo_index.xml and later used at runtime. As psql handles them natively quite fine, the extra information from dmo_index is ignored.

Because all sequence functionality is based on int64 datatype, this update contains the int64 patches.
Also, this update contains the runtime for sequences for both psql and h2 (the latter with minor glitches that will be solved with #1911).

I know that this is not "by the book" and it's difficult to review/test as it was for me to clean/extract the patch. The problem is that on one hand, this issue depends on int64 implementation and, on the other hand, I could not implement the conversion without knowing exactly how the runtime will work and what data/resources will be needed. In the previous update I tried to separate them by providing only the stub runtime functions and it failed to compile :(.

In @34 note above, Eric said that the following issues have occurred:

- sequence names need to be converted to SQL-compliant names using the NameConversionWorker;
- there was a latent defect in DataModelWorker.Library.sortClasses, which would drop all sequences for a schema with more than one table.

and that he fixed them. I do not have those fixes, so they may need to be reapplied.

**#45 - 01/10/2013 01:28 AM - Eric Faulhaber**

*- File om_upd20130109b.zip added*

Code review 20130109a:

Good work overall! However, I did have one serious problem; see below.

I made some minor changes; please replace your local version of om_upd20130109a.zip with om_upd20130109b.zip accordingly:

- p2o.xml: update number in p2o.xml header is 039 instead of 037.
- hibernate_templates.tpl: removed "T" column from header; corrected comment for DMO index sequence element.
- generate_ddl.xml: removed empty "JPRM" column from header.
- DataModelTokenTypes.java: removed "T" column from header.
- ImportWorker.java: minor format change.
- ConversionDriver.java: minor format changes.
- MathOps.java: minor format changes (fixed strange spacing between methods).
- DMOIndex.java: changed copyright year; corrected some javadoc.
- SequenceManager.java: removed "JPRM" column from header; minor format changes.
- SequenceHandler class hierarchy: removed "JRPM" column from header; minor format changes, javadoc corrections.
- PsqlSequenceHandler.java: renamed class to PostgreSQLSequenceHandler to be more consistent with other dialect-specific areas of the code.
- Sequence.java: minor format changes.

Question: in dmo-index.xml, do we want the sequence's name to be the historical, Progress name, or the converted, SQL-safe name (I guess the latter)? Don't make any change for this at this point; I will do it if needed when I apply my other changes for the sequence name conversion. I just need to know, since this dmo-index part is a new feature relative to my original fix.

Some other issues:

- int64.java: I noted there are a number of TODOs in the code. Some are missing functionality, some look like they will cause runtime problems (e.g., ClassCastException). However, I agree that it's best to leave these for the companion, runtime support task, which is lower priority right now.
- ...and the main problem: ~~MathOps~~ [correction: Operators] doesn't compile:

```
    [javac] /mnt/san/sata/gc/20121029/p2j/src/com/goldencode/p2j/persist/pl/Operators.java:1415: incompatible
types
    [javac] found   : java.lang.Long
    [javac] required: java.lang.Integer
    [javac]          return MathOps.modulo(new decimal(a), new decimal(b)).toJavaType();
    [javac]                                                                ^
    [javac] /mnt/san/sata/gc/20121029/p2j/src/com/goldencode/p2j/persist/pl/Operators.java:1437: incompatible
types
    [javac] found   : java.lang.Long
    [javac] required: java.lang.Integer
    [javac]          return MathOps.modulo(new decimal(a), new integer(b)).toJavaType();
    [javac]                                                               ^
    [javac] /mnt/san/sata/gc/20121029/p2j/src/com/goldencode/p2j/persist/pl/Operators.java:1459: incompatible
types
    [javac] found   : java.lang.Long
    [javac] required: java.lang.Integer
    [javac]          return MathOps.modulo(new integer(a), new decimal(b)).toJavaType();
    [javac]                                                               ^
```

Please fix this last issue (based on the 0109b update) and resubmit so we can regression test. This is **high priority**, since our main regression test environment is currently broken because of this error.

#### #46 - 01/10/2013 08:06 AM - Ovidiu Maxiniuc

*- File om_upd20130110a.zip added*

Apparently I omitted to put com.goldencode.p2j.persist.pl.Operators.java file on the update package.
I also re-review and did some minor format changes.

#### #47 - 01/14/2013 11:36 AM - Eric Faulhaber

No code review comments.  We put this update through regression testing and it has passed.  Please re-confirm these changes do not conflict with any other recent updates in bzr.  If not, it should be committed and distributed.  If so, please do not commit and let me know.  Please do this ASAP.

#### #48 - 01/14/2013 11:56 AM - Ovidiu Maxiniuc

I just updated a fresh copy from bzr repository and checked all modified files. There are no conflicts.
I am going now to commit the changes and distribute the update package.

#### #49 - 01/14/2013 04:06 PM - Eric Faulhaber

*- Status changed from Review to Closed*

*- % Done changed from 90 to 100*

#### #50 - 04/25/2013 10:45 AM - Eric Faulhaber

*- Estimated time changed from 16.00 to 126.00*

#### #51 - 11/16/2016 09:30 AM - Greg Shah

*- Target version deleted (Milestone 3)*

#### #52 - 11/16/2016 10:52 AM - Greg Shah

*- Target version set to Conversion Support for Server Features*

### Files

| | | | |
|---|---|---|---|
| om_upd20121210a.zip | 354 KB | 12/10/2012 | Ovidiu Maxiniuc |
| om_upd20121210b.zip | 32.9 KB | 12/10/2012 | Ovidiu Maxiniuc |
| om_upd20121211a.zip | 387 KB | 12/11/2012 | Ovidiu Maxiniuc |
| om_upd20121214a.zip | 387 KB | 12/14/2012 | Ovidiu Maxiniuc |
| om_upd20130109a.zip | 516 KB | 01/09/2013 | Ovidiu Maxiniuc |
| om_upd20130109b.zip | 516 KB | 01/10/2013 | Eric Faulhaber |
| om_upd20130110a.zip | 522 KB | 01/10/2013 | Ovidiu Maxiniuc |