Database - Feature #2134

configurable denormalization of extent fields into individual fields in schema conversion

04/25/2013 01:37 PM - Eric Faulhaber

Status:	Closed	**Start date:**	08/13/2013
Priority:	Normal	**Due date:**	08/23/2013
Assignee:	Vadim Nebogatov	**% Done:**	100%
Category:		**Estimated time:**	60.00 hours
Target version:	Cleanup and Stablization for Server Features		
billable:	No	**vendor_id:**	GCD

Description

Related issues:

Related to Database - Feature #2376: retain bulk getter/setter methods for cu...	**Test**
Related to Database - Feature #2580: make field denormalization the default s...	**New**

History

#1 - 05/02/2013 01:55 PM - Eric Faulhaber

- Estimated time set to 60.00

The idea is to make the normalization of extent fields into separate database tables configurable, on an all-or-nothing basis. In the case these fields are not normalized, they will be converted to array types.

This is going to require the following changes:

- do not composite fields of like extent during schema conversion;
- map extent fields to array types instead of lists in the Hibernate ORM;
- generate the DMO implementation classes differently (use different internal variable types, get rid of composite, inner classes);
- I think we can leave the interfaces unchanged, unless we need the ability to get an entire array at once -- we don't need this for converted code, but it may be a customer requirement (potential data corruption concerns here);
- possible changes to the persistence runtime where there is special code dealing with collections (maybe this can be left in place, I guess it just won't be called anymore).

Not sure what else may need changes at the moment...

IIRC, there were some limitations in how Hibernate could deal with arrays; need some investigation to determine if this will get in our way.

#2 - 05/02/2013 04:48 PM - Eric Faulhaber

- Due date set to 08/23/2013

- Start date set to 08/13/2013

- Assignee set to Eric Faulhaber

#3 - 10/17/2013 08:56 AM - Eric Faulhaber

- Assignee changed from Eric Faulhaber to Vadim Nebogatov

The requirements of this task have changed somewhat.

The change will no longer take place on an all or nothing basis; instead, we need to develop a new type of schema hint which specifies a particular extent field to be expanded to a series of individual fields (NOT to a SQL array type). A convenience version of the hint should allow a table to be specified instead of a field. All extent fields in that table would be so expanded.

Composite inner classes will still be in use for those extent fields which are not so hinted, so this change is additive, not destructive.

We need a default naming conversion for extent fields which are so expanded (e.g., field(1)-->field_1, field(2)-->field2, ... field(N)-->fieldN). Note that we may need to adjust this piece or make it configurable, depending on customer preferences (e.g., allowing individual indexes to be named explicitly, as in, hours(1)-->monday_hours, hours(2)-->tuesday_hours, etc.).

Name changes should be done early enough in schema conversion that they flow through normal name conversion process naturally.

#4 - 10/17/2013 09:00 AM - Eric Faulhaber

- Subject changed from configurable denormalization of extent fields into arrays in schema conversion to configurable denormalization of extent fields into individual fields in schema conversion

#5 - 10/31/2013 05:49 AM - Vadim Nebogatov

Issue status.

Found out and tested all places affecting custom extent conversion. Studied and tested hints generation, in particular, schema/table hints. New hint element <custom-extent> is created. New hint creation and parsing in progress; hint looks like

```
<hints>
   <schema>

      <!--   custom-extent hint for all extents in table -->
      <table name="table1">
         <custom-extent/>
      </table>

      <!--   custom-extent hints for concrete fields in table for default naming conversion for extent fields
which are so expanded (e.g., field(1)-->field_1,     field(2)-->field2, ... field(N)-->fieldN) -->
      <table name="table2">
         <custom-extent>
            <field name="field1"/>
            <field name="field2"/>
         </custom-extent>
      </table>

      <!--   custom-extent hints for concrete fields in table for configurable, depending on customer preferen
ces naming conversion (e.g., allowing individual indexes to be named explicitly, as in, hours(1)-->monday_hour
s, hours(2)-->tuesday_hours, etc.). -->
      <table name="table3">
         <custom-extent>
            <field name="field1">
               <extent-names>field1First,field1Second,field1Third</extent-names>
            </field>
         </custom-extent>
      </table>
   </schema>
</hints>
```

#6 - 11/01/2013 06:03 PM - Vadim Nebogatov

Parsing result is field in TableHints

```
   /** Map containing information about custom extent conversion */
private Map<String, List <String>> extents = null;
```

Corresponding parsing code in TableHints constructor is

```
      ...
      // Read custom extent elements.
      list = table.getElementsByTagName(ELEM_CUSTOM_EXTENT);
      len = list.getLength();
      if (len > 0)
      {
         extents = new HashMap<String, List <String>>();
         for (int i = 0; i < len; i++)
         {
            Element extent = (Element) list.item(i);
            NodeList fieldsList = extent.getElementsByTagName(ELEM_FIELD);
            int fieldsListLen = fieldsList.getLength();
            if (fieldsListLen == 0)
            {
               extents.clear();
               break;
            }
            else
            {
               for (int k = 0; k < fieldsListLen; k++)
               {
                  Element field = (Element) fieldsList.item(k);
                  NodeList extentNames = field.getElementsByTagName(ELEM_EXTENT_NAMES);
                  List<String> fieldNames;
                  if (extentNames.getLength() == 0)
                  {
                     fieldNames = new ArrayList<String>();
                  }
                  else
                  {
                     Element extentNamesElement = (Element) extentNames.item(0);
                     String names = extentNamesElement.getTextContent();
                     fieldNames = StringHelper.parseTokens(names, ",", true, true, true);
                  }
                  String fieldName = field.getTextContent();
                  extents.put(fieldName, fieldNames);
               }
            }
         }
      }
      ...
```

I think one method would be enough for using in rules:

```
   public List<String> getCustomExtent(String fieldName)
   {
      if (extents == null)
      {
         return null;
      }
      else if (extents.size() == 0)
      {
         return new ArrayList<String>();
      }
      else
      {
         return extents.get(fieldName);
      }
   }
```

According to parsing algorithm, it returns

```
1) null, if field has no custom extent processing,
2) empty list, if field has auto-increment based extent processing
3) non-empty properties list for using in custom extent processing
```

Rules files where this method should be used

```
1) p2o.xml
2) iface_props.rules
3) dmo_common.rules
4) dmo_common.rules
5) hibernate.xml
```

All places related with

```
<rule>extentVal != null
...
```

#7 - 11/06/2013 10:01 AM - Eric Faulhaber

Vadim, what are you thinking in terms of how to deal with business logic that references former extent fields which have been expanded using these new hints? Currently, we call the specialized (i.e., indexed) forms of the DMO getter and setter methods (e.g., dmo.getSomething(1) or dmo.setSomething(1, newValue)).

As I see it, there are 2 use cases to deal with in this regard:

- references with static/constant indexes which are known at conversion time; these can be converted to the new, simple (i.e., unindexed) getters/setters;
- references with indexes which are not known at conversion time (runtime variables or expressions).

The second case should be much less common, but it is much more complicated to deal with, as we need some way to correlate the specific extent field element with the new, expanded field.

Thoughts?

#8 - 11/07/2013 08:48 AM - Vadim Nebogatov

I think we should generate class implementing CompositeUserType for each such property both for incremented names and custom names.
It should look like MoneyUserType class in http://javadata.blogspot.no/2011/07/hibernate-and-user-types.html and will allow both storing in different columns in database and using specified property names in HQL queries.

#9 - 11/07/2013 08:55 AM - Vadim Nebogatov

This interface is already used in DatetimeTzUserType.

#10 - 11/07/2013 01:20 PM - Eric Faulhaber

Help me understand why we would need a custom Hibernate type for every extent field in an application. We already have basic extent field support working with the existing conversion. With the hints you are implementing, we are changing references to specific elements within an extent field to instead reference a specific DMO property. Whereas today, extent fields reside in secondary tables associated by foreign key to a primary table, the new schema structure would expand/flatten an extent field such that each element would be stored as a separate column in the primary table. Where do custom user types come into it?

#11 - 11/07/2013 02:15 PM - Vadim Nebogatov

I think that in any case we need some way to enumerate elementary properties specified in hint in runtime to provide index based getter/setter access. I have no experience with CompositeUserType, but it seems to me it could be useful because it allows using such access like it is done in MoneyUserType. Besides it also allows storing each property value in a separate column of the primary table.

#12 - 11/07/2013 04:19 PM - Eric Faulhaber

I think the common case we will encounter will be statically indexed references to extent fields, which we can resolve at conversion. So, for example, today we convert the following 4GL:

```
schedule.daily-hours[1] = 8.
schedule.daily-hours[2] = 6.
```

to (roughly):

```
schedule.setDailyHours(0, new integer(8));
schedule.setDailyHours(1, new integer(6));
```

...and in the event a hint specifies that the schedule table's daily-hours extent 5 field should be expanded to schedule.monday-hours, schedule.tuesday-hours, etc., the converted code should look something like this:

```
schedule.setMondayHours(new integer(8));
schedule.setTuesdayHours(new integer(6));
```

So, the index parameter is dropped from the dmo property setter, and because of the hint, we know that daily-hours[1] now means monday-hours, daily-hours[2] now means tuesday-hours, etc. This can all be determined at conversion time, because of the static/constant references to the extent field indexes in the code.

The case for which we would need to enumerate the properties specified in the hint at runtime would be the presumably less common case of the value of an extent field's index not being known until runtime (in a variable or runtime expression). For example:

```
def var i as int init 1.
schedule.daily-hours[i] = 8.
```

I say "presumably less common" because it is likely that the extent fields which are candidates for this hint-based expansion are those that don't particularly map well into extent fields today, and which the customer would wish to map into individual fields, given the chance. Because they aren't

"natural" fits for extent fields, I expect they are probably used more like regular fields, so the references to them are likely less dynamic and more static. This of course is just an educated guess, we'll see whether it is proven by the actual code we see.

In any case, we need a solution for these dynamic index references, regardless of how rare or common they are. So, the question is, how do we handle this second case? Probably still like this:

```
integer i = new integer(1);
schedule.setDailyHours(minus((i), 1), new integer(8));
```

So, we still have a getter and setter method pair in that DMO interface for the original, daily-hours extent 5 field (i.e., getDailyHours(int index) and setDailyHours(int index, NumberType dailyHours)). At runtime, we internally map (via the RecordBuffer DMO proxy) any remaining invocations of the legacy, indexed operation (due to variable/expression index references) to the correct, backing property in the DMO and table.

The preferred way (in my opinion) to convey the legacy information about which new DMO property goes with which old extent field index would be with annotations on the DMO implementation class, derived from the schema hint at schema conversion.

There is also the matter of the sizer DMO methods (e.g., see sizeHours in the Person dmo in the p2j_test schema) and the array-based getters and setters (e.g., integer[] getHours and void setHours(integer[]), also in Person). Again, I think we resolve these in the RecordBuffer DMO proxy and omit them from the DMO implementation class altogether. I've actually been wanting to do this for a while.

As a final, polish step (after the core functionality is working), I think we should make the emission of all these legacy, extent-related methods in the DMO interface conditional upon whether they are even in use in the application. Better if we can eliminate this baggage altogether. This can be determined during schema/annotations.xml, when we sweep the application code looking for other schema-related information. Perhaps we make this step configuration-dependent, because I can see some customers wanting to leave that API in place, in case there are external dependencies on the schema which would still require them.

But these last two steps are for later cleanup anyway. Let's get the basics working first.

Start with the common case of static/constant index references. This should mostly be a matter of converting the schema structure from an extent field to multiple, individual fields early in the process, so that the rest of the schema conversion happens naturally. I think even data import should happen without any programming changes, as long as the individual fields are expanded in the order of their related extent elements.

The best place for these changes probably is early in the p2o.xml rule sets, before any of the composite, inner class computations are done. But leave annotations behind in those individual fields so we have the information necessary later on to add back the legacy indexed methods in the DMO interface.

The next step will be to tackle the variable/expression index references, etc.

Does this sound like a workable set of goals and rough plan? Please ask any questions you may have.

#13 - 11/07/2013 04:38 PM - Greg Shah

Two thoughts:

> ...and in the event a hint specifies that the schedule table's daily-hours extent 5 field should be expanded to schedule.monday-hours, schedule.tuesday-hours, etc., the converted code should look something like this:

I assume that we also would support a scenario where someone has set a hint to denormalize an extent field BUT has not given us a replacement naming scheme. In such a case, I would expect that the names would look like this:

```
schedule.setDailyHours0(new integer(8));
schedule.setDailyHours1(new integer(6));
```

Right?

> As a final, polish step (after the core functionality is working), I think we should make the emission of all these legacy, extent-related methods in the DMO interface conditional upon whether they are even in use in the application. Better if we can eliminate this baggage altogether. This can be determined during schema/annotations.xml, when we sweep the application code looking for other schema-related information. Perhaps we make this step configuration-dependent, because I can see some customers wanting to leave that API in place, in case there are external dependencies on the schema which would still require them.

I don't think this is possible in applications that use dynamic database. You would have to inspect all possible dynamic query strings to determine if these fields could be referenced. I don't think it is worth the effort.

#14 - 11/07/2013 05:21 PM - Eric Faulhaber

Greg Shah wrote:

> I assume that we also would support a scenario where someone has set a hint to denormalize an extent field BUT has not given us a replacement naming scheme. In such a case, I would expect that the names would look like this:
>
> [...]
>
> Right?

Right, there should be a default naming convention like this.

> As a final, polish step...

> I don't think this is possible in applications that use dynamic database. You would have to inspect all possible dynamic query strings to determine if these fields could be referenced. I don't think it is worth the effort.

Yes, good point. This would not be possible with apps in which dynamic queries are data- or user-driven.

#15 - 11/11/2013 01:42 PM - Vadim Nebogatov

I have created loop in p2o.xml in rule with the comment

<!-- map fields to properties -->

after rule with the comment

<!-- get field level hints for this table -->

As result several properties are generated instead of one with extent. But I cannot find the place where they should be initialized: for example DMO constructor for Person looks like

```
public PersonImpl()
{
   siteId = new integer(0);
   ...
   extension = new character("");
   schedule1;
   schedule2;
   schedule3;
   schedule4 = new character("");
   hours1;
   hours2;
   hours3 = new integer(0);
}
```

Only latest property is initialized. Values 3 and 4 are just for test so far.

#16 - 11/11/2013 02:54 PM - Eric Faulhaber

Look at the .p2o file that is generated and compare the artificial property nodes you have created with your expansion hints to a regular, "organic" property node. I suspect the initial sub-node is missing from all but one of your artificial nodes, indicating that your expansion is creating partial copies/clones of the original, extent field property node. Make sure each artificial node has all the same content as an organic property node (plus whatever annotations you need downstream to indicate what happened).

The instance variable initializers are created in dmo_props.rules, which is driven by java_dmo.xml. The initializer rules rely on each property node having a sub-node of type data.initial. Look for the comment "set initial value" in the dmo_props rule-set.

#17 - 11/11/2013 04:51 PM - Vadim Nebogatov

Yes, initial subnode was missing, thanks.

#18 - 11/14/2013 10:20 AM - Vadim Nebogatov

- File vmn_upd20131114a.zip added

- File vmn_upd20131114b.zip added

Attached

1) code update vmn_upd20131114a.zip merged with latest bzr revision 10411;
2) vmn_upd20131114b.zip containing some examples and conversion results: hints file, changed for test p2j_test.df file with added some more fields with extents and some conversion results.

It is not final: proper annotations are still required and there are some errors like

<property column="hours9" name="hours3" type="p2j_integer"/>

in PersonImpl.hbm.xml.

#19 - 11/14/2013 04:07 PM - Vadim Nebogatov

Issue with

<property column="hours9" name="hours3" type="p2j_integer"/>

is resolved.

#20 - 11/15/2013 05:47 PM - Vadim Nebogatov

Issue is found and fixed concerning wrong conversion of fields with no extent in case if hint is assigned for whole table.
One more issue is found: DMO getters/setters are generated properly, but converted code still uses wrong (index based) getters. For example,

```
person.extension[3] = "test3"
person.extension[4] = "test4"
```

converted to

```
person.setExtension6(2, new character("test3"));
person.setExtension6(3, new character("test4"));
```

Field extension with extent = 6.

#21 - 11/17/2013 06:08 AM - Vadim Nebogatov

- File vmn_upd20131117a.zip added

- File vmn_upd20131117b.zip added

Updated files attached

#22 - 11/20/2013 06:29 PM - Vadim Nebogatov

Regresion 20131119_150620 executed with 3 FAILED tests:

1. tc_job_002 FAILED failure in step 40: 'Unexpected EOF in actual at page # 1.'
2. tc_item_master_104 FAILED failure in step 45: 'Mismatched data at absolute row 52415, relative
row 41, page # 904, page size 58, last page false, column index 20. Expected 'A' (0x41) and found 'B' (0x42).'
3. tc_pay_sfc_003 FAILED failure in step 55: 'timeout before the specific screen buffer became available (Mismatched data at line 21, column 0. Expected 'R' (0x0052 at relative Y 21, relative X 0) and found '' (0x0000 at relative Y 21, relative X 0).)'

Regression started one more time.

#23 - 11/21/2013 05:59 PM - Eric Faulhaber

Code review 20131117*:

This is a good start, but it only handles the schema conversion, DMO creation, and ORM configuration side of the issue. The next step is to deal with the business logic conversion as documented above. Note that we'll have to do something a little different than leaving the sizer and indexed getter/setter methods in the interfaces while omitting them from the implementation classes, because the implementation classes implement the interfaces. We'll have to think about that a little bit, but the rest of the plan above should be ready.

With regard to the hint implementation:

- Why did you choose a CSV format for the <extent-names> hint element, instead of using separate <extent-name> elements for each name?
- Is there a check to ensure that the number of "hinted" names provided matches the number of elements in the original extent field? I may have misunderstood the changes to p2o.xml, but I didn't see that.

#24 - 11/21/2013 06:24 PM - Vadim Nebogatov

Second regresion 20131121_123331 executed with 6 FAILED tests:

1. tc_job_002 FAILED failure in step 40: 'Unexpected EOF in actual at page # 1.'

2. tc_dc_slot_024 FAILED failure in step 8: 'timeout before the specific screen buffer became available (Mismatched data at line 4, column 9. Expected '' (0x0000 at relative Y 4, relative X 9) and found '1' (0x0031 at relative Y 4, relative X 9).)'
3. tc_dc_slot_029 FAILED failure in step 20: 'timeout before the specific screen buffer became available (Mismatched data at line 4, column 0. Expected '' (0x0000 at relative Y 4, relative X 0) and found 'L' (0x004C at relative Y 4, relative X 0).)'
4. tc_dc_slot_024 FAILED failure in step 8: 'timeout before the specific screen buffer became available (Mismatched data at line 4, column 9. Expected '' (0x0000 at relative Y 4, relative X 9) and found '1' (0x0031 at relative Y 4, relative X 9).)'
5. tc_job_clock_004 FAILED failure in step 3: 'Timeout while waiting for event semaphore to be posted.'
6. tc_pay_emp_abs_acc_003 FAILED failure in step 26: 'timeout before the specific screen buffer became available (Mismatched data at line 21, column 0. Expected 'W' (0x0057 at relative Y 21, relative X 0) and found 'O' (0x004F at relative Y 21, relative X 0).)'

So, regression is passed.

#25 - 11/22/2013 07:01 AM - Eric Faulhaber

Go ahead and commit and distribute the update, even though the above issues need to be addressed. It won't affect conversion or runtime unless people actually use the hint, so it should be fine.

#26 - 11/22/2013 02:41 PM - Vadim Nebogatov

Eric Faulhaber wrote:

- Why did you choose a CSV format for the <extent-names> hint element, instead of using separate <extent-name> elements for each name?

I thought about this, hint based <extent-names> seemed shorter. But I agree that separate <extent-name> elements look better. Should I correct?

- Is there a check to ensure that the number of "hinted" names provided matches the number of elements in the original extent field? I may have misunderstood the changes to p2o.xml, but I didn't see that.

There is no such check so far, only TODO in TableHints:

```
TODO: check in conversion time sizes of this list and extent value
```

I was not sure if this check is needed (runtime error will be thrown in any case if extent elements size less than extent value), how correctly throw error/exception from TRPL and if conversion process should be stopped after such error.

#27 - 11/22/2013 03:08 PM - Eric Faulhaber

Vadim Nebogatov wrote:

> [...] But I agree that separate <extent-name> elements look better. Should I correct?

Yes, please, but as part of the further work. A separate update now for just this is not necessary.

> > Is there a check to ensure that the number of "hinted" names provided matches the number of elements in the original extent field? [...]
>
> I was not sure if this check is needed (runtime error will be thrown in any case if extent elements size less than extent value), how correctly throw error/exception from TRPL and if conversion process should be stopped after such error.

If there are fewer hinted names than the original extent, we can use the default naming strategy for the remainder and print a warning during conversion with printfln (you should be able to find other examples of this). If there are more hinted names than the original extent, this seems like a genuine error. Use throwException in this case (See CommonAstSupport$Library.throwException).

Please note that since this is for M11, it is lower priority work than any M7 issue you have.

#28 - 11/29/2013 02:45 PM - Vadim Nebogatov

Note 27 implemented in #2126, note 49

#29 - 12/02/2013 09:12 AM - Vadim Nebogatov

For getters/setters recognition in business logic, I suggest to add “index” property to LegacyField. It will be filled only for fields with custom extents conversion hints:

```
public @interface LegacyField
{
   /** The name of the legacy field associated with a DMO property. */
   String name();

   /**
    * Index in the legacy extent field associated with a DMO property. It will be filed only
    * for field with custom extents.
    */
   int index() default -1;
}
```

DMO annotations will look like:

```
...
   /** Ext */
   @LegacyField(name = "extension")
   private final character extension;

   /** Data member */
   @LegacyField(name = "schedule",  index = 1)
   private final character schedule2;

   /** Data member */
   @LegacyField(name = "schedule",  index = 2)
   private final character schedule1;

   /** Data member */
   @LegacyField(name = "schedule",  index = 3)
   private final character scheduleThird;

   /** Data member */
   @LegacyField(name = "schedule",  index = 4)
   private final character scheduleFourth;

   /** Data member */
   @LegacyField(name = "schedule",  index = 5)
   private final character scheduleFifth;

   /** Hours */
   @LegacyField(name = "hours",  index = 1)
   private final integer hours1;

   /** Hours */
   @LegacyField(name = "hours",  index = 2)
   private final integer hours2;

   /** Hours */
   @LegacyField(name = "hours",  index = 3)
   private final integer hours3;
...
```

#30 - 12/04/2013 04:36 PM - Eric Faulhaber

This approach makes sense to me.

#31 - 12/04/2013 05:08 PM - Vadim Nebogatov

Question: the problem now is that we corrected *.p2o for fields with custom extent conversion, but code conversion parser uses original, not corrected *.dict file. What do you think about similar changes in *.dict file? Could it give some negative effect? I see that *.dict files are not processed now and reflects *.df file.

#32 - 12/04/2013 05:16 PM - Eric Faulhaber

The *.dict file is used for parsing/interpreting the original 4GL code (as opposed to generating converted code), so I don't think this is necessary, but perhaps I don't understand your idea. Please clarify what specific problem(s) you want to address by adjusting the *.dict file.

#33 - 12/04/2013 05:59 PM - Eric Faulhaber

To expand on my last post, the place where the DMO getter and setter method names are determined for purposes of converted code is rules/annotations/record_scoping_post.rules (look for calls to javaGetterName and javaSetterName). It is here you will need to do the switch to the customized names -- IF we have a static array index value -- and hide/remove that static array index value so it does not emit as a parameter to the renamed getter/setter method. You will have to make the P2OAccessWorker$Library.java{G|S}etterName methods more sophisticated, to account for the extent value when determining the correct getter/setter name. You may need to make changes in core conversion (i.e., rules/convert/...) as well.

In the event of a variable extent value in business logic, you still will have to use the old, indexed getter/setter method names, as discussed above in note 12. This is where you will need the LegacyField index annotations at runtime.

#34 - 12/04/2013 07:02 PM - Vadim Nebogatov

I tried the way you ponted, namely with changes in javaGetterName and javaSetterName methods. Initial problem was how to pass field index there, but I noticed later that we can take this index from emitted child of LBRACKET type (and will have to remove LBRACKET nodes after that). Another one problem so far for using this approach is how to pass correct "fieldname", now it is generated as latest name from custom properties name specified in hint.

On parsing time, SymbolResolver uses SchemaDictionary based on not changed schema file, so in business_file.p.ast I see some old field properties like in *.dict file, for example "extent". So I thought maybe some other places also will be affected with "old" "extent" property, possible WHERE clause generation or something else.

#35 - 12/05/2013 01:33 PM - Eric Faulhaber

Yes, you will need to track down all the places in code conversion we use the extent value to make some analysis/annotation decision or to emit code differently, and make sure these use the correct name.

The uses that come to mind are direct references to fields in 4GL logic (reading or writing a field), and in database access (i.e., FIND, FOR, OPEN QUERY, dynamic queries, ...). The latter is primarily about WHERE clauses. AFAIR, you cannot define an index to use a specific extent field element, though IIRC, you may be able to use them in BY and BREAK BY clauses, which would affect the sort clauses we generate. Please confirm.

#36 - 12/10/2013 04:35 PM - Vadim Nebogatov

In *.ast file, for expression like

person.schedule[1] = person.schedule[4].

"EXPRESSION" ast node missed for right operand.

For left operand:

```
<ast col="19" id="12884901939" line="7" text="[" type="LBRACKET">
  <ast col="0" id="12884901940" line="0" text="expression" type="EXPRESSION">
    <annotation datatype="java.lang.Long" key="peerid" value="180388626507"/>
    <ast col="20" id="12884901941" line="7" text="1" type="NUM_LITERAL">
      <annotation datatype="java.lang.Boolean" key="use64bit" value="false"/>
      <annotation datatype="java.lang.String" key="java_value" value="0"/>
      <annotation datatype="java.lang.Long" key="peerid" value="180388626508"/>
    </ast>
  </ast>
</ast>
```

For right operand:

```
  <ast col="40" id="12884901945" line="7" text="[" type="LBRACKET">
    <ast col="41" id="12884901946" line="7" text="4" type="NUM_LITERAL">
      <annotation datatype="java.lang.Boolean" key="use64bit" value="false"/>
      <annotation datatype="java.lang.String" key="java_value" value="3"/>
      <annotation datatype="java.lang.Long" key="peerid" value="180388626511"/>
    </ast>
  </ast>
```

I expected it should like similar. Are there any reason of this?

#37 - 12/10/2013 04:57 PM - Greg Shah

I know it is confusing, but this does occur for a reason. The top-most node of an expression is always EXPRESSION. Any recursion within the expression itself does not repeat the EXPRESSION node. Having this EXPRESSION node at the top is necessary to allow us to find and process the root of all expressions in the project.

There do exist cases where elements that most commonly appear in expressions can also appear outside of an expression. An lvalue is the most

common case of this. Normal lvalues (on the left side of an assignment operator) are not part of an expression, so they don't have any EXPRESSION node. BUT the LBRACKET case is an exception since it must be able to have an arbitrarily complex expression as the index. In this case, the index expression will have an EXPRESSION node.

There are many places in the TRPL rules where we have to handle this "quirk". Unfortunately, I don't know of a better way to solve the problem (it is the lesser of two evils).

#38 - 12/11/2013 06:24 PM - Vadim Nebogatov

Statically indexed references seems converted correctly now, for example

```
person.schedule[5] = person.schedule[4] + person.schedule[1] + person.schedule[2].
```

converted using custom names from hints to

```
person.setScheduleFifth(concat(person.getScheduleFourth(), person.getSchedule2(), person.getSchedule1()));
```

What will we do now with dynamic references for custom extents? Recognize them on record_scoping_post.rules processing and throw exception with message like “Dynamic references could not be converted with custom names for fields with extent”?

#39 - 12/11/2013 11:46 PM - Eric Faulhaber

Vadim Nebogatov wrote:

> What will we do now with dynamic references for custom extents? Recognize them on record_scoping_post.rules processing and throw exception with message like “Dynamic references could not be converted with custom names for fields with extent”?

Well, I think there is some merit to this approach, because if a customer explicitly is making the change to separate fields, dynamic references in the code aren't really appropriate anymore. However, the current thinking is that they should convert as they do today, in case they are invoked (possibly unintentionally or in a forgotten place -- like a dynamic query) in an application. Nevertheless, I think we should log a warning during conversion, so these can be cleaned up by the customer if desired.

Please re-read note 12 above, as well as the follow-up discussion, which (partially) addresses this topic. For now, the plan is to leave the indexed getter and setter methods in the DMOs. All dynamic calls in code will be handled by the dynamic proxy already created in RecordBuffer. Dynamic references in HQL will be a bit trickier -- not sure of the best way to deal with these yet.

#40 - 12/12/2013 06:38 AM - Greg Shah

And there may be dynamic-extent references in dynamic-database queries. I don't think these will all be cleaned up. Nor will it be possible to detect them all at conversion time. It just means we must be able to map from the un-normalized legacy names to the custom names at runtime.

#41 - 12/12/2013 12:06 PM - Eric Faulhaber

The HQL is the problem with the dynamic references. After the new names are used in the DMOs and the Hibernate mappings, we can no longer execute HQL queries which expect the old mappings (i.e., to a secondary table instead of to the primary table). Some fairly complex HQL preprocessing could be added to rewrite the queries to reference the new names, but only if we can calculate the extent field index at runtime, before the query is submitted. If the index is read from the database, however, we won't know how to rewrite the HQL (i.e., which new name/field to reference).

#42 - 12/17/2013 01:41 PM - Vadim Nebogatov

On 12/16/2013 05:28 PM, Vadim wrote:
Eric, Greg,

Are there any examples or manual how to correctly emit KW_SWITCH element or create corresponding java AST template with CASEs/BREAKs inside?

Thanks,
Vadim

On 17.12.2013 03:17, Eric Faulhaber wrote:
Vadim,

Your question makes me worry. <g>

I can't think of any reason you would need to emit a switch statement in converted code, in connection with this task. Please update #2134 with a description/example of what you are planning to do. I suspect there has been some miscommunication, which I would like to clear up as quickly as possible.

Thanks,
Eric

On 12/17/2013 04:47 AM, Vadim wrote:
Eric,

I need switch to implement indexed access for field with extent with custom name conversion.

For example, field "schedule" extent 2 has hints:

```
<field name="schedule">
  <extent-name>scheduleFirst</extent-name>
  <extent-name>scheduleSecond</extent-name>
</field>
```

I expect to see the following code in corresponsing *Impl.java file:

```
@LegacyField(name = "schedule",  index = 1)
private final character scheduleFirst;

@LegacyField(name = "schedule",  index = 2)
private final character scheduleSecond;

public character getScheduleFirst()
{
  return (new character(scheduleFirst));
}

public void setScheduleFirst(Text scheduleFirst)
{
  this.scheduleFirst.assign(scheduleFirst, true);
}

public character getScheduleSecond()
{
  return (new character(scheduleSecond));
```

```
  }

  public void setScheduleSecond(Text scheduleSecond)
  {
    this.scheduleSecond.assign(scheduleSecond, true);
  }

  public void setSchedule(int index, character schedule)
  {
    switch (index)
    {
      case 1:
        setScheduleFirst(schedule);
        break;
      case 2:
        setScheduleSecond(schedule);
        break;
      default:
        throw new IllegalArgumentException("Wrong index for setSchedule()");
    }
  }

  public character getSchedule(int index)
  {
    switch (index)
    {
      case 1:
        return getScheduleFirst();
      case 2:
        return getScheduleSecond();
      default:
        throw new IllegalArgumentException("Wrong index for getSchedule()");
    }
  }
```

Thanks,
Vadim

On 17.12.2013 21:50, Eric Faulhaber wrote:
Vadim,

Ah, I see where you are going, but the intention was not to do it this way, as it adds code to every affected DMO implementation class for what is essentially an obsolete feature. Rather, the intent was to do the "switching" inside the dynamic proxy implementation within RecordBuffer. I was expecting we would simply throw UnsupportedOperationException from these methods in the implementation class and annotate them as @deprecated. Eventually, I'd like to remove the methods from the implementation classes altogether and refactor the DMO interfaces such that implementations of these methods are not necessary at all, but this is more effort than I want to tackle right now.

BTW, please move this discussion into Redmine.

Thanks,
Eric

#43 - 12/18/2013 05:03 PM - Vadim Nebogatov

I have created in RecordBuffer

```
private final Map<Method, List<Method>> customExtentMap;
```

mapping indexed setters/getters to custom methods.

Also I added to RecordBuffer

```
private final class BufferHandler implements InvocationHandler
```

which method invoke() executes using proxy handler (like now) if method is not key from customExtentMap otherwise executes corresponding method from this map by index. There will be no switch at all in such case. Am I on the correct way?

The question is how to fill customExtentMap. What do you think if add annotation for additional indexed methods based on hints like

```
@LegacyMethod(
{
   @LegacyCustomMethod(name = "setFirst"),
   @LegacyCustomMethod(name = "setSecond"),
   @LegacyCustomMethod(name = "setThird"),
   ...
})
```

As result, customExtentMap will be just filled with methods having such annotations.

#44 - 12/18/2013 05:24 PM - Eric Faulhaber

I think your approach is good. If you can cleanly refactor this feature into a stand-alone, top-level class with minimal change inside the RecordBuffer class itself, that would be even better, because RecordBuffer already is so cluttered with functionality. I'd like to prevent it from growing further where possible.

#45 - 12/26/2013 06:21 AM - Vadim Nebogatov

Metadata problem is found for denormalized fields with extent:

Caused by: org.h2.jdbc.JdbcSQLException: Unique index or primary key violation: "IDX__META_FIELD_FIELD_POSITION ON PUBLIC.META_FIELD(FILE_RECID, ORDER_)

Error reason is that order has the same value for all custom fields for given field with extent.

Is this unique index really needed? Or I can assign new orders for fields in DMO declared below denormalized field(s)? For example, with some shift.

How is supposed to use denormalized fields with extent if some screen is based on such field and order? Could such orders change give some runtime problems?

#46 - 01/26/2014 06:26 PM - Vadim Nebogatov

Could you check my last question (note 45) about order? Similar question should be addressed with newly added fieldId. Current LegacyFieldInfo is storage for denormalized fields, but several LegacyFieldInfo instances (with different fieldId) should correspond to one denormalized field with extent.

#47 - 01/27/2014 11:23 AM - Eric Faulhaber

The unique index is needed; it is part of the legacy metadata schema and an application may include code which relies on this index existing. I think you can assign new values for order. This denormalization is a departure from legacy behavior that a customer knowingly undertakes, and so the denormalized fields won't work exactly as before from a metadata perspective (i.e., will not have the old value for order). However, all the other tables that do not contain denormalized fields, as well as the other fields in a table that contains some denormalized field, must work as before, so we cannot make structural changes to the metadata schema, like removing an index or the uniqueness of an index. I think a change in order with a commensurate shift (so that the relative sequence remains intact) is the best we can do. Same thing for fieldId.

#48 - 01/30/2014 03:52 PM - Vadim Nebogatov

- File vmn_upd20140130a.zip added

I have attached vmn_upd20140130a.zip with first tests results containing examples of generated DMOs, table hints and converted business files.
Do we need to execute some more complicated tests?
Order and fieldId issues are still in progress (I simply removed corresponding unique index so far).

#49 - 02/04/2014 05:48 PM - Vadim Nebogatov

- File vmn_upd20140204a.zip added

I have attached vmn_upd20140204a.zip for your review.

Updated with latest bzr revision 10458.

Test cases were attached earlier as vmn_upd20140130a.zip.

#50 - 02/06/2014 05:24 PM - Eric Faulhaber

Thank you for providing the tests and especially the hints and conversion output; it is very helpful for doing the review.

Vadim Nebogatov wrote:

> Do we need to execute some more complicated tests?

Yes. These tests access the extent fields directly from 4gl code, which is good, but we also need to support extent field references within a WHERE clause, both with constant and variable indexes. Both static (e.g., FIND, FOR, etc.) and dynamic queries (e.g., CREATE QUERY) should be tested. This will require that you become familiar with the HQLPreprocessor class, since it is here that we will have to fix up HQL WHERE clauses.

But first, please focus on the following two paragraphs:

Why is the @CustomExtentMethod annotation needed? Don't all the getter and setter methods simply follow a JavaBean naming convention, where a set or get/is prefix is prepended to the capitalized variable name? Doesn't the @LegacyField annotation on the instance variable give us enough information to follow this deterministic setter/getter naming convention without the @CustomExtentProperty and @CustomExtentMethod annotations?

I think it would be useful to enhance the hint syntax further to allow an optional description and label for denormalized fields. When a developer/DBA is re-purposing extent fields and assigning new names to the individual elements, it is very likely the extent field members will have entirely new purposes/meanings. The description would be used for javadoc on the variable and getter/setter methods and the label would be used for the label member of the @LegacyField annotation. The latter change would modify the behavior of the application, but that is pretty much the intent of this feature, so a customer's acceptance testing would have to make exceptions for this behavior. If these specifications are missing from the hint, the default behavior would apply.

I have not finished reviewing the code update, but I wanted to provide this feedback without further delay.

#51 - 02/06/2014 05:29 PM - Eric Faulhaber

Eric Faulhaber wrote:

> This will require that you become familiar with the HQLPreprocessor class, since it is here that we will have to fix up HQL WHERE clauses.

I should clarify: for constant indexes, given that the schema changes will already be in place when converting 4gl queries referencing denormalized extent fields, it may be that we can handle this in conversion rules. It is the variable index cases where we will need to do some more sophisticated runtime HQL preprocessing.

#52 - 02/07/2014 08:22 AM - Vadim Nebogatov

Eric Faulhaber wrote:

> Why is the @CustomExtentMethod annotation needed? Don't all the getter and setter methods simply follow a JavaBean naming convention, where a set or get/is prefix is prepended to the capitalized variable name? Doesn't the @LegacyField annotation on the instance variable give us enough information to follow this deterministic setter/getter naming convention without the @CustomExtentProperty and @CustomExtentMethod annotations?

CustomExtentProperty annotation needed for remember denormalized property name, this “pseudo” property is absent in DMO.

CustomExtentMethods – I think you are right. I created it in time when LegacyField had no required information. I will try to remove this annotation at all.

#53 - 02/07/2014 11:52 AM - Eric Faulhaber

Vadim Nebogatov wrote:

> CustomExtentProperty annotation needed for remember denormalized property name, this “pseudo” property is absent in DMO.

I'm still not clear why this is needed, because the name within the CustomExtentProperty annotation (e.g., "hours" or "schedule" in the PersonImpl example) can be deterministically derived from the method name, again using the JavaBean naming convention. Is it more problematic to do it this way? Does using the annotation save us some storage overhead at runtime?

#54 - 02/07/2014 05:20 PM - Vadim Nebogatov

I added CustomExtentProperty as indicator that property was denormalized. I thought it was simpler. In principle, I agree, we can analyze getDeclaredMethods() for DMO and compare it with getDeclaredFields() in order to find all denormalized properties (corresponding properties will have no declared fields). Do you mean such way? I think it will be not big overhead.

#55 - 02/07/2014 05:31 PM - Vadim Nebogatov

From other side, it is even not needed: such properties already exists in LegacyField annotation:

```
   /** Data member */
   @LegacyField( fieldId = 16 , name = "schedule,1" , index = 1)
   private final character schedule2;

   /** Data member */
   @LegacyField( fieldId = 17 , name = "schedule,2" , index = 2)
   private final character schedule1;

   /** Data member */
   @LegacyField( fieldId = 18 , name = "schedule,3" , index = 3)
```

```
  private final character scheduleThird;
```

...

#56 - 02/22/2014 05:40 PM - Vadim Nebogatov

- File vmn_upd20140222a.zip added

- File vmn_upd20140222b.zip added

I have attached next update vmn_upd20140222a.zip with using LegacyField instead of earlier using CustomExtentProperty/CustomExtentMethod annotations and added optional description and label for denormalized fields. vmn_upd20140222b.zip contains examples of generated DMO, table hints and converted business file.

#57 - 02/25/2014 12:41 PM - Eric Faulhaber

Code review 20140222a/b:

This update looks good overall, though I think you've changed more than you needed to (see point about API changes below). I like the cleaner hint syntax, and the converted test code looks good. I do have a few concerns and questions:

- I'm not sure the metaschema.xml changes are desirable. The metaschema database is something the legacy code relies on to report on the legacy schema. Legacy code will not be expecting queries against this database to report any information about the denormalized schema; I think that is likely to break things. What was the purpose of these changes?
- There is a new throw_exception template in java_templates.tpl. I didn't see where this was used. Is it in use?
- You changed some public APIs in P2OAccessWorker and P2OLookup, adding an index parameter. Rather than changing the existing API and fixing up all the places that use it, please add the new API, but leave the old one (which represents the more common use case) in place. The old one should call the new one with a sensible default value for the new parameter. Existing callers of the old API should remain unchanged. This should reduce the scope of your update, since it appears in some files, changing callers to use the new APIs was the only change. Please follow this idiom in general when dealing with existing APIs, unless there is a compelling reason not to.
- New method P2OLookup.addJavaName has no javadoc.
- The change in DMOValidator is only to the header and to the formatting of one line, so the header entry description doesn't match the change to the file. Please double-check that there wasn't something more you had intended to do with this file. If not, back out this change.
- The header entry to LegacyField says: ** 004 VMN 20140222 Added index() and original(), and ends there, in the middle of the sentence. I see a new index method, but nothing called original.
- Does the new method PropertyHelper.getPropertyName(Method) need to be public instead of package private? If so, please group it in the file with other public methods. Why not re-use the makePropertyName method from within this new method?
- TableMapper.getPropertyName needs javadoc.
- You have made changes to very sensitive portions of RecordBuffer (the invocation handler and detectChange, in particular). Regression testing will be very important for this update. Please look carefully at the server.log for any anomalies, even if testing passes.

#58 - 02/26/2014 09:00 AM - Eric Faulhaber

Vadim, please ignore my first question in the code review about the metaschema changes. I forgot we had discussed that above. Once you address

the remaining issues in the code review, please submit the update to regression testing and in parallel move on to addressing the query-related issues in this task.

#59 - 02/26/2014 03:00 PM - Vadim Nebogatov

Eric Faulhaber wrote:

- There is a new throw_exception template in java_templates.tpl. I didn't see where this was used. Is it in use?

It is used in dmo_common.rules:

```
tpl.graft('throw_exception', null, peerNodeId,
   'except', 'UnsupportedOperationException')
```

for generation of indexed getter/setters for denormalized field like

```
   public Text getSchedule(int index)
   {
      throw new UnsupportedOperationException();
   }
```

- You changed some public APIs in P2OAccessWorker and P2OLookup, adding an index parameter. Rather than changing the existing API and fixing up all the places that use it, please add the new API, but leave the old one (which represents the more common use case) in place. The old one should call the new one with a sensible default value for the new parameter. Existing callers of the old API should remain unchanged. This should reduce the scope of your update, since it appears in some files, changing callers to use the new APIs was the only change. Please follow this idiom in general when dealing with existing APIs, unless there is a compelling reason not to.

Ok, good.

- New method P2OLookup.addJavaName has no javadoc.

Added.

- The change in DMOValidator is only to the header and to the formatting of one line, so the header entry description doesn't match the change to the file. Please double-check that there wasn't something more you had intended to do with this file. If not, back out this change.

It happened because earlier there were some other changes in this file related with index. This file has no changes now.

- The header entry to LegacyField says: ** 004 VMN 20140222 Added index() and original(), and ends there, in the middle of the sentence. I see a new index method, but nothing called original.

Typo is fixed ("," is replaced to "."). original is in the end of class:

```
   /**
    * Original property name (actually not exposed in DMO) for denormalized field with
    * extent.
    */
   String original() default "";
```

- Does the new method PropertyHelper.getPropertyName(Method) need to be public instead of package private? If so, please group it in the

file with other public methods. Why not re-use the makePropertyName method from within this new method?

I thought it could be used in some more places. Now it could be package private. makePropertyName is reused.

- TableMapper.getPropertyName needs javadoc.

Fixed.

- You have made changes to very sensitive portions of RecordBuffer (the invocation handler and detectChange, in particular). Regression testing will be very important for this update. Please look carefully at the server.log for any anomalies, even if testing passes.

Ok. good.

#60 - 02/28/2014 04:05 AM - Vadim Nebogatov

- File vmn_upd20140227a.zip added

I have attached next update vmn_upd20140227a.zip with fixes according note 57 and refactored classes P2OLookup, P2OAccessWorker and TableMapper (old public API is preserved). Test cases and test results were not changes since last update.

Updated with latest bzr revision 10483.

#61 - 03/02/2014 02:34 PM - Vadim Nebogatov

I have found the following conversion error (see log below) in current bzr revision for the following example:

```
def var i as int.
i = 3.
find first person where person.schedule[i] = person.schedule[i + 1].
```

Some another example is converted successfully:

```
def var i as int.
i = 3.
find first person where person.schedule[i] = "qwerty".
```

EXPRESSION EXECUTION ERROR:

---

ext1 = create("java.lang.Integer", ref.text)

^ { java.lang.reflect.InvocationTargetException }

Elapsed job time: 23:00:01.590

ERROR:

java.lang.RuntimeException: ERROR! Active Rule:

RULE REPORT

Rule Type : ASCENT

Source AST: [=] BLOCK/STATEMENT/KW_FIND/RECORD_PHRASE/KW_WHERE/EXPRESSION/EQUALS/ @3:44 {12884901924}

Copy AST : [=] BLOCK/STATEMENT/KW_FIND/RECORD_PHRASE/KW_WHERE/EXPRESSION/EQUALS/ @3:44 {12884901924}

Condition : ext1 = create("java.lang.Integer", ref.text)

Loop : false

--- END RULE REPORT ---

at com.goldencode.p2j.pattern.PatternEngine.run(PatternEngine.java:863)

at com.goldencode.p2j.convert.ConversionDriver.processTrees(ConversionDriver.java:931)

at com.goldencode.p2j.convert.ConversionDriver.back(ConversionDriver.java:819)

at com.goldencode.p2j.convert.ConversionDriver.main(ConversionDriver.java:1756)

Caused by: com.goldencode.expr.ExpressionException: Expression execution error @1:8 [EQUALS id=12884901924]

at com.goldencode.p2j.pattern.AstWalker.walk(AstWalker.java:226)

at com.goldencode.p2j.pattern.AstWalker.walk(AstWalker.java:160)

at com.goldencode.p2j.pattern.PatternEngine.apply(PatternEngine.java:1350)

at com.goldencode.p2j.pattern.PatternEngine.processAst(PatternEngine.java:1248)

at com.goldencode.p2j.pattern.PatternEngine.processAst(PatternEngine.java:1196)

at com.goldencode.p2j.pattern.PatternEngine.run(PatternEngine.java:832)

... 3 more

Caused by: com.goldencode.expr.ExpressionException: Expression execution error @1:8

at com.goldencode.expr.Expression.execute(Expression.java:434)

at com.goldencode.p2j.pattern.Rule.apply(Rule.java:401)

at com.goldencode.p2j.pattern.Rule.executeActions(Rule.java:640)

at com.goldencode.p2j.pattern.Rule.coreProcessing(Rule.java:609)

at com.goldencode.p2j.pattern.Rule.apply(Rule.java:440)

at com.goldencode.p2j.pattern.Rule.executeActions(Rule.java:640)

at com.goldencode.p2j.pattern.Rule.coreProcessing(Rule.java:609)

at com.goldencode.p2j.pattern.Rule.apply(Rule.java:440)

at com.goldencode.p2j.pattern.Rule.executeActions(Rule.java:640)

at com.goldencode.p2j.pattern.Rule.coreProcessing(Rule.java:609)

at com.goldencode.p2j.pattern.Rule.apply(Rule.java:440)

at com.goldencode.p2j.pattern.Rule.executeActions(Rule.java:640)

at com.goldencode.p2j.pattern.Rule.coreProcessing(Rule.java:609)

at com.goldencode.p2j.pattern.Rule.apply(Rule.java:440)

at com.goldencode.p2j.pattern.RuleContainer.apply(RuleContainer.java:531)

at com.goldencode.p2j.pattern.RuleSet.apply(RuleSet.java:50)

at com.goldencode.p2j.pattern.AstWalker.ascent(AstWalker.java:249)

at com.goldencode.ast.AnnotatedAst$1.notifyListenerLevelChanged(AnnotatedAst.java:2195)

at com.goldencode.ast.AnnotatedAst$1.hasNext(AnnotatedAst.java:2100)

at com.goldencode.p2j.pattern.AstWalker.walk(AstWalker.java:207)

... 8 more

Caused by: java.lang.reflect.InvocationTargetException

at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)

at sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:57)

at sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.java:45)

at java.lang.reflect.Constructor.newInstance(Constructor.java:526)

at com.goldencode.p2j.pattern.CommonAstSupport$Library.create(CommonAstSupport.java:589)

at com.goldencode.p2j.pattern.CommonAstSupport$Library.create(CommonAstSupport.java:512)

at com.goldencode.expr.CE3354.execute(Unknown Source)

at com.goldencode.expr.Expression.execute(Expression.java:341)

... 27 more

Caused by: java.lang.NumberFormatException: For input string: "i"

at java.lang.NumberFormatException.forInputString(NumberFormatException.java:65)

at java.lang.Integer.parseInt(Integer.java:492)

at java.lang.Integer.<init>(Integer.java:677)

... 35 more

#62 - 03/12/2014 07:43 AM - Vadim Nebogatov

Status of where clause conversion for denormalized extent fields

Problem was in what time should be denormalized fields in WHERE clause replaced with new property values. After analyzing and debugging I came to conclusion that converted WHERE clause should look like without denormalization

```
"upper(person.schedule[?]) = ?"
```

and actual replacement will be done in HQLPreprocessor.preprocess(...) method immediately before analyzeComposites() invocation:

```
         Set<HQLAst> denormalizedFields = getDenormalizedFields(root);
         if (denormalizedFields != null)
         {
            for (HQLAst property : denormalizedFields)
            {
               inlineDenormalizedField(property, parameters);
            }

            if (LOG.isTraceEnabled())
            {
               String sep = System.getProperty("line.separator");
               LOG.trace("[" + database + "] Denormalized HQL:  " + hql + sep + root.dumpTree());
            }
         }

         Map<HQLAst, String> targets = analyzeComposites(root);
         ...
```

As you can see denormalized HQL already contains schedule2 property instead of schedule and has no LBRACKET child:

```
[03/12/2014 14:39:13 MSK] (com.goldencode.p2j.persist.HQLPreprocessor:FINEST) [local/p2j_test/primary] Origina
l HQL:  upper(person.schedule[?]) = ?
= [EQUALS] @1:27
   upper [FUNCTION] @1:1
      person [ALIAS] @1:7
         schedule [PROPERTY] @1:14
            [ [LBRACKET] @1:22
               ? [SUBST] @1:23
   ? [SUBST] @1:29

[03/12/2014 14:39:13 MSK] (com.goldencode.p2j.persist.HQLPreprocessor:FINEST) [local/p2j_test/primary] Denorma
lized HQL:  upper(person.schedule[?]) = ?
= [EQUALS] @1:27
   upper [FUNCTION] @1:1
      person [ALIAS] @1:7
         schedule2 [PROPERTY] @1:14
   ? [SUBST] @1:29
```

New method HQLPreprocessor.getDenormalizedFields uses

```
(TableMapper.getLegacyFieldInfo(dmoClass, property.getText()) == null)
```

as criteria for recognition of denormalized property (schedule in our case) :

```
   private Set<HQLAst> getDenormalizedFields(HQLAst root)
   {
      Set<HQLAst> denormalizedFields = null;

      Iterator<HQLAst> iter = root.iterator();
      while (iter.hasNext())
      {
         HQLAst next = iter.next();
```

```
         HQLAst alias = null;
         HQLAst property = null;
         HQLAst subscript = null;

         // Do we have an alias node?
         if (next.getType() == ALIAS)
         {
            alias = next;
            property = (HQLAst) next.getChildAt(0);

            // Does the alias have a property child?
            if (property != null && property.getType() == PROPERTY)
            {
               // Look up DMO implementation class which corresponds with
               // alias.
               RecordBuffer buffer = resolveBuffer(alias);
               Class<?> dmoClass = buffer.getDMOImplementationClass();

               if (TableMapper.getLegacyFieldInfo(dmoClass, property.getText()) == null)
               {
                  // Look for an LBRACKET, which indicates a possible composite.
                  HQLAst lbracket = (HQLAst) property.getChildAt(0);

                  // Does the property have a subscript?
                  if (lbracket != null && lbracket.getType() == LBRACKET)
                  {
                     subscript = (HQLAst) lbracket.getChildAt(0);
                  }
               }
               else
               {
                  property = null;
               }
            }
            else
            {
               property = null;
            }
         }
         else
         {
            alias = null;
         }

         if (alias == null || property == null || subscript == null)
         {
            continue;
         }

         if (denormalizedFields == null)
         {
            denormalizedFields = new HashSet<>();
         }

         denormalizedFields.add(property);
      }

      return denormalizedFields;
   }
```

New method HQLPreprocessor. InlineDenormalizedField(...) replaces denormalized fields with new properties based on extentIndex calculated from passed parameter value:

```
   // not clean variant
   private void inlineDenormalizedField(HQLAst property, Object[] parameters)
         throws AstException
   {
      HQLAst subst = (HQLAst) property.getChildAt(0).getChildAt(0);

      String dataType = (String) subst.getAnnotation("datatype");
      int index = ((Long) subst.getAnnotation("index")).intValue();
      Object parm = parameters[index];
```

```
      if (parm instanceof Resolvable && !(parm instanceof FieldReference))
      {
         // Do not inline Resolvables.  If we get here, it represents a
         // programming error.  Warn, but don't fail.
         if (LOG.isErrorEnabled())
         {
            StringBuilder buf =
                  new StringBuilder("Resolvable substitution parameter @");
            buf.append(index);
            buf.append(" (");
            buf.append(parm);
            buf.append(") cannot be inlined [");
            buf.append(hql);
            buf.append("]");
            LOG.error(buf.toString(), new Throwable());
         }

         return;
      }

      HQLTypes hqlType = Enum.valueOf(HQLTypes.class, dataType);
      int extentIndex;
      switch (hqlType)
      {
         case INTEGER:
            extentIndex = ((integer) parm).intValue();
            break;
         case LONG:
            extentIndex = ((int64) parm).intValue();
            break;
         default:
            return;
      }

      String denormalizedProperty = TableMapper.getDenormalizedProperty(dmoClass, property.getText(), extentIn
dex);
      property.setText(denormalizedProperty);
      property.removeChildren();
      property.putAnnotation("inlined", Boolean.TRUE);
      inline = true;
   }
```

One more change needed in where_clause_normalize.rules:

```
...
   <worker class="com.goldencode.p2j.schema.P2OAccessWorker"
           namespace="p2o" />
...
   <variable name="schemaname1"    type="java.lang.String" />
   <variable name="schemaname2"    type="java.lang.String" />
...
         <!-- for comparisons between fields which have passed the first level
              of screening, we now have to check whether we are dealing with
              extents;  if so, we can only reduce the subexpression if both
              fields are extent fields AND they are dereferencing the same
              index -->
         <rule>result and evalLib("fieldtype", c1.type)
            <action>
               schemaname1 = #(java.lang.String) c1.getAnnotation("schemaname")
            </action>
            <action>
               schemaname2 = #(java.lang.String) c2.getAnnotation("schemaname")
            </action>
            <rule>(c1.isAnnotation("extent") and not isDenormalizedProperty(schemaname1)) or
                  (c2.isAnnotation("extent") and not isDenormalizedProperty(schemaname2))
             ...
```

Test cases I used so far like

```
def var i as int.
i = 2.
find first person where person.schedule[1] = person.schedule[4].
if available person then message "FOUND".
for each person where person.schedule[1] = person.schedule[4]:
  delete person.
end.
```

They passed now both in conversion and runtime (last results are still in testing).

I tried also test like

```
def var i as int.
i = 2.
find first person where person.schedule[i] = person.schedule[i + 1].
```

but such test seems gives conversion error in latest revision (note 61), could you check if this is real bug?

#63 - 03/12/2014 07:06 PM - Vadim Nebogatov

UnsupportedOperationException with using WhereExpression found for denormalized fields with extent. Test case uses field in subscript – it works in 4GL

```
find first person where person.schedule[person.emp-num - (i modulo 3) - 172 - 222 + 2] = "aa".
```

Corresponding converted fragment:

```
   WhereExpression whereExpr0 = new WhereExpression()
   {
      public logical evaluate(final BaseDataType[] args)
      {
         return isEqual(person.getSchedule(minus((plus(minus(minus(minus(person.getEmpNum(), modulo(i, 3)), 17
2), 222), 2)), 1)), "aa");
      }
   };
```

#64 - 03/12/2014 08:24 PM - Eric Faulhaber

WhereExpression itself does not throw UnsupportedOperationException. What is throwing it? What is your question?

#65 - 03/12/2014 08:42 PM - Eric Faulhaber

- File ecf_upd20140312b.zip added

The attached update fixes the conversion error reported in note 61. I have not yet regression tested it.

#66 - 03/13/2014 03:21 AM - Vadim Nebogatov

Eric Faulhaber wrote:

> WhereExpression itself does not throw UnsupportedOperationException. What is throwing it? What is your question?

UnsupportedOperationException is thrown with indexed version of schedule getter. Dynamic calls in code are handled by the dynamic proxy created in RecordBuffer. I think we should use proxy for WhereExpression too.

#67 - 03/14/2014 01:30 AM - Vadim Nebogatov

Problem resolved. Replacement original method to denormalized method was missed in temporary record mode in RecordBuffer. Now test with WhereExpression passed.

#68 - 03/15/2014 01:33 PM - Vadim Nebogatov

Concerning bug in where_clause_normalize.rules (note 61).

Solution would be to add check ref.type == prog.num_literal before cast ref.text to Integer:

```
        <rule>result and evalLib("fieldtype", c1.type)
            <rule>c1.isAnnotation("extent") or c2.isAnnotation("extent")
               <action on="false">result = true</action>
               <action>ext1 = null</action>
               <action>ext2 = null</action>
               <rule>c1.isAnnotation("extent")
                  <action>ref = c1.getChildAt(0).getChildAt(0)</action>
                  <rule>ref.type == prog.num_literal
                     <action>ext1 = create("java.lang.Integer", ref.text)</action>
                  </rule>
               </rule>
               <rule>c2.isAnnotation("extent")
                  <action>ref = c2.getChildAt(0).getChildAt(0)</action>
                  <rule>ref.type == prog.num_literal
                     <action>ext2 = create("java.lang.Integer", ref.text)</action>
                  </rule>
               </rule>
               <action>result = (ext1 != null) and (ext1 == ext2)</action>
            </rule>
         </rule>
```

#69 - 03/17/2014 03:31 AM - Vadim Nebogatov

Current problem: denormalized extent fields with constant index in WHERE clause are generated without index but with old property name.

#70 - 03/17/2014 11:20 AM - Eric Faulhaber

Vadim Nebogatov wrote:

> Concerning bug in where_clause_normalize.rules (note 61).
>
> Solution would be to add check ref.type == prog.num_literal before cast ref.text to Integer:
>
> [...]

This is what I submitted in note 65. Did you see it?

#71 - 03/17/2014 05:11 PM - Vadim Nebogatov

Eric Faulhaber wrote:

> Vadim Nebogatov wrote:
>
> > Concerning bug in where_clause_normalize.rules (note 61).
> >
> > Solution would be to add check ref.type == prog.num_literal before cast ref.text to Integer:
> >
> > [...]
>
> This is what I submitted in note 65. Did you see it?

Unfortunately I missed your update... But I am happy that I was on the correct way.

#72 - 03/17/2014 07:03 PM - Eric Faulhaber

Vadim Nebogatov wrote:

> Unfortunately I missed your update...

Please maintain this fix with the rest of your changes for this task. I will not plan to regression test it separately.

#73 - 03/18/2014 02:37 AM - Vadim Nebogatov

Eric Faulhaber wrote:

> Vadim Nebogatov wrote:
>
> > Unfortunately I missed your update...
>
> Please maintain this fix with the rest of your changes for this task. I will not plan to regression test it separately.

OK. Sorry again. I recently changed Internet provider because of my local relocation and have constant problems with sending emails and submit notes to Redmine (sometimes 10-20 times to send). I think I was one again busy with sending and missed your update.

#74 - 03/18/2014 03:14 PM - Vadim Nebogatov

- File vmn_upd20140318b.zip added

- File vmn_upd20140318a.zip added

I have attached next update vmn_upd20140318a.zip with WHERE clause support. Update corresponds to latest bzr revision 10493.

vmn_upd20140318b.zip contains examples of generated DMO, table hints and converted business file. I tested business file with commenting some lines.

One difference is found so far between 4GL and P2J behavior for CREATE QUERY

```
create query h.
h:add-buffer(buffer person:handle).
l = h:query-prepare("for each person where person.schedule[1] = 'aa'") no-error.
h:query-open().
```

Both 4GL and P2J pass this test, but P2J runtime gives additional message

```
QUERY-OPEN for query  requires a previous QUERY-PREPARE. (7312)
```

#75 - 03/21/2014 06:23 AM - Vadim Nebogatov

- File vmn_upd20140320b.zip added

I have attached some more tests vmn_upd20140320b.zip with WHERE clause support.

#76 - 03/27/2014 03:42 AM - Vadim Nebogatov

- File vmn_upd20140327a.zip added

I have attached latest source code changes vmn_upd20140327a.zip after some fixes during regression. Regression started again.

#77 - 03/28/2014 06:32 AM - Vadim Nebogatov

- File vmn_upd20140327a.zip added

I have attached vmn_upd20140328a.zip after some fixes during regression. Latest regression results: 2 tests failed.

```
1. tc_job_002 FAILED failure in step 40: 'Unexpected EOF in actual at page # 1.'
2. tc_job_matlcron_001 FAILED failure in step 46: 'java.lang.RuntimeException: java.lang.RuntimeException: 2:
No such file
        at com.goldencode.harness.test.FileComparison.execute(FileComparison.java:144)
        at com.goldencode.harness.test.Test.execute(Test.java:278)
        at com.goldencode.harness.Driver.run(Driver.java:121)
Caused by: java.lang.RuntimeException: 2: No such file
        at com.goldencode.harness.transport.SFTPSession.get(SFTPSession.java:284)
        at com.goldencode.harness.test.FileComparison.download(FileComparison.java:190)
        at com.goldencode.harness.test.FileComparison.execute(FileComparison.java:129)
        at com.goldencode.harness.test.Test.execute(Test.java:278)
        at com.goldencode.harness.Driver.run(Driver.java:121)
Caused by: com.jcraft.jsch.SftpException: No such file
        at com.jcraft.jsch.ChannelSftp.throwStatusError(ChannelSftp.java:2793)
        at com.jcraft.jsch.ChannelSftp._stat(ChannelSftp.java:2149)
        at com.jcraft.jsch.ChannelSftp._stat(ChannelSftp.java:2166)
        at com.jcraft.jsch.ChannelSftp.get(ChannelSftp.java:878)
        at com.jcraft.jsch.ChannelSftp.get(ChannelSftp.java:838)
        at com.goldencode.harness.transport.SFTPSession.get(SFTPSession.java:278)
        at com.goldencode.harness.test.FileComparison.download(FileComparison.java:190)
        at com.goldencode.harness.test.FileComparison.execute(FileComparison.java:129)
        at com.goldencode.harness.test.Test.execute(Test.java:278)
        at com.goldencode.harness.Driver.run(Driver.java:121)'
```

Second failure seems not relevant to changes.

#78 - 03/28/2014 06:35 AM - Vadim Nebogatov

- File deleted (vmn_upd20140327a.zip)

#79 - 03/28/2014 06:36 AM - Vadim Nebogatov

- File vmn_upd20140328a.zip added

#80 - 03/28/2014 06:29 PM - Vadim Nebogatov

Next regression completed. Again 2 tests failed. Second test is another one but with the same error:

```
1. tc_job_002 FAILED failure in step 40: 'Unexpected EOF in actual at page # 1.'

2. gso_145 FAILED failure in step 49: 'java.lang.RuntimeException: java.lang.RuntimeException: 2: No such file
        at com.goldencode.harness.test.FileComparison.execute(FileComparison.java:144)
        at com.goldencode.harness.test.Test.execute(Test.java:278)
        at com.goldencode.harness.Driver.run(Driver.java:121)
Caused by: java.lang.RuntimeException: 2: No such file
        at com.goldencode.harness.transport.SFTPSession.get(SFTPSession.java:284)
        at com.goldencode.harness.test.FileComparison.download(FileComparison.java:190)
        at com.goldencode.harness.test.FileComparison.execute(FileComparison.java:129)
        at com.goldencode.harness.test.Test.execute(Test.java:278)
        at com.goldencode.harness.Driver.run(Driver.java:121)
Caused by: com.jcraft.jsch.SftpException: No such file
        at com.jcraft.jsch.ChannelSftp.throwStatusError(ChannelSftp.java:2793)
        at com.jcraft.jsch.ChannelSftp._stat(ChannelSftp.java:2149)
        at com.jcraft.jsch.ChannelSftp._stat(ChannelSftp.java:2166)
        at com.jcraft.jsch.ChannelSftp.get(ChannelSftp.java:878)
        at com.jcraft.jsch.ChannelSftp.get(ChannelSftp.java:838)
        at com.goldencode.harness.transport.SFTPSession.get(SFTPSession.java:278)
        at com.goldencode.harness.test.FileComparison.download(FileComparison.java:190)
        at com.goldencode.harness.test.FileComparison.execute(FileComparison.java:129)
        at com.goldencode.harness.test.Test.execute(Test.java:278)
        at com.goldencode.harness.Driver.run(Driver.java:121)
```

#81 - 03/28/2014 07:24 PM - Eric Faulhaber

tc_job_002 is expected to fail in step 40. The other two failing tests each passed in one of your runs, so you can consider this update as having passed regression testing. Please commit and distribute.

#82 - 03/29/2014 10:21 AM - Vadim Nebogatov

- File vmn_upd20140329a.zip added

Attached update is passed regression and distributed.

#83 - 03/29/2014 12:17 PM - Vadim Nebogatov

- File deleted (vmn_upd20140329a.zip)

#84 - 03/29/2014 12:24 PM - Vadim Nebogatov

- File vmn_upd20140329a.zip added

#85 - 03/30/2014 03:08 PM - Eric Faulhaber

- % Done changed from 0 to 70

- Status changed from New to WIP

In order to call this task complete, we need some documentation on how the new feature is used. Please add a history entry here (not a separate document) describing:

- the hint syntax;
- the practical impact of denormalizing an extent field on the DDL, the converted code, and runtime behavior;
- limitations;
- anything else you believe is relevant.

The intended audience of this documentation is a developer involved in converting an application, but not necessarily someone involved in developing the conversion technology. We eventually will integrate this into the Conversion Handbook and/or Conversion Reference.

#86 - 04/03/2014 10:31 AM - Vadim Nebogatov

Configurable denormalization of extent fields into individual fields

Two types of field denormalization are supported:

1. default denormalization - default naming conversion for extent fields which are so expanded (e.g., field(1)-->field_1, field(2)-->field2, ... field(N)-->fieldN)
2. custom denormalization - configurable, depending on customer preferences (e.g., allowing individual indexes to be named explicitly, as in, hours(1)-->monday_hours, hours(2)-->tuesday_hours, etc.).

Conversion uses table hints for custom denormalization of extent fields.

The hint syntax.

Table denormalization hint is described using <custom-extent>, <field> and <extent-field> elements:

```
<hints>
  <schema>
       <table>
           <custom-extent>
                <field>
                    <extent-field>
```

<custom-extent> element has no attributes;
<field> element has one attributes: name;
<extent-field> element has 3 attributes: name, label and description.

Denormalization could be set both on table and field level. If <custom-extent> node has no children, all extent fields will be converted with default denormalization. Otherwise only children fields will be denormalization corresponding <field> nodes. If field node has no children, corresponding field will be converted with default denormalization otherwise will be converted with custom denormalization according to <extent-field> children. Other extent fields will be not denormalized.

Example:

```
<hints>
   <schema>

      <!-- custom-extent hint for all extent fields in table -->
      <table name="vehicle">
         <custom-extent/>
      </table>

      <!-- custom-extent hints for concrete fields-->
      <table name="person">
         <custom-extent>

            <!-- configurable custom naming conversion -->
            <field name="schedule">
               <extent-field name = "schedule2" label = "label of schedule2 text"/>
               <extent-field name = "schedule1" description = "description of schedule1 text"/>
               <extent-field name = "scheduleThird" label = "label of scheduleThird text" description = "descr
iption of scheduleThird text"/>
               <extent-field name = "scheduleFourth"/>
               <extent-field name = "scheduleFifth"/>
            </field>

            <!-- default naming conversion-->
           <field name="hours"/>

         </custom-extent>
      </table>
   </schema>
</hints>
```

Corresponding extent fields are declared in p2j_test.df file:

```
...
ADD FIELD "schedule" OF "Person" AS character
  FORMAT "x(64)"
  INITIAL ""
  SQL-WIDTH 650
  EXTENT 5
  ORDER 160

ADD FIELD "hours" OF "Person" AS integer
  FORMAT "99"
  INITIAL "0"
  LABEL "Hours"
  SQL-WIDTH 30
  EXTENT 5
  ORDER 170
...
ADD FIELD "options" OF "vehicle" AS character
  DESCRIPTION "Optional features"
  FORMAT "x(32)"
  INITIAL ""
  LABEL "Options"
  SQL-WIDTH 660
  HELP "Enter an optional feature"
  EXTENT 10
  ORDER 50
...
```

In this example

- table vehicle - all fields with extent will have default denormalization
- table person - field schedule will have custom denormalization, field hours will have default naming conversion.

The practical impact of denormalizing an extent field on the DDL, the converted code, and runtime behavior.

DDL changes.

Nexw fields will be used in schema_table_*.sql:

```
    create table person (
        ...
        schedule2 varchar,
        schedule1 varchar,
        schedule_third varchar,
        schedule_fourth varchar,
        schedule_fifth varchar,
        hours1 integer,
        hours2 integer,
        hours3 integer,
        hours4 integer,
        hours5 integer,
        ...
    );
```

```
    create table vehicle (
        ...
        options1 varchar,
        options2 varchar,
        options3 varchar,
        options4 varchar,
        options5 varchar,
        options6 varchar,
        options7 varchar,
        options8 varchar,
        options9 varchar,
        options10 varchar,
        ...
    );
```

DMO changes

LegacyField annotation field name attribute contains absent property name like without denormalization, it is used for indexed methods support. Also added original attribute containing legacy field name and index attribute containing corresponding index.

Along with new properties getters/setters, DMO contains indexed getters/setters corresponding to (absent) initial property name.

Converted code changes

Indexed getters/setters of extent field with constant indexes are replaced with corresponding getters/setters of new fields. Other indexed getters/setters of extent field will be not changed. In runtime execute of such methods will be converted to execute of new getters/setters with using special proxy in RecordBuffer.

Runtime behavior

Support extent field references within a WHERE clause. Converted WHERE clause looks like without denormalization, actual replacement is done in runtime in HQLPreprocessor.preprocess(...) method. As result denormalized HQL will contain replaced new fields.

#87 - 04/04/2014 09:52 AM - Eric Faulhaber

- % Done changed from 70 to 100

- Status changed from WIP to Closed

#88 - 08/15/2014 04:00 PM - Vadim Nebogatov

Test cases are checked in to testcases/uast/denormalization folder.

#89 - 02/05/2015 11:16 AM - Stanislav Lomany

- File svl_upd20150205a.zip added

Update that excludes temporary table fields from denormalization process. Committed to bzr revision 10743.

#90 - 11/16/2016 12:06 PM - Greg Shah

- Target version changed from Milestone 11 to Cleanup and Stablization for Server Features

Files

vmn_upd20131114a.zip	22.3 KB	11/14/2013	Vadim Nebogatov
vmn_upd20131114b.zip	13.1 KB	11/14/2013	Vadim Nebogatov
vmn_upd20131117a.zip	22.3 KB	11/17/2013	Vadim Nebogatov
vmn_upd20131117b.zip	40.3 KB	11/17/2013	Vadim Nebogatov
vmn_upd20140130a.zip	10.2 KB	01/30/2014	Vadim Nebogatov
vmn_upd20140204a.zip	274 KB	02/04/2014	Vadim Nebogatov
vmn_upd20140222a.zip	272 KB	02/22/2014	Vadim Nebogatov
vmn_upd20140222b.zip	5.11 KB	02/22/2014	Vadim Nebogatov
vmn_upd20140227a.zip	179 KB	02/28/2014	Vadim Nebogatov
ecf_upd20140312b.zip	3.93 KB	03/13/2014	Eric Faulhaber
vmn_upd20140318a.zip	214 KB	03/18/2014	Vadim Nebogatov
vmn_upd20140318b.zip	28.6 KB	03/18/2014	Vadim Nebogatov
vmn_upd20140320b.zip	56.3 KB	03/21/2014	Vadim Nebogatov
vmn_upd20140327a.zip	214 KB	03/27/2014	Vadim Nebogatov
vmn_upd20140328a.zip	214 KB	03/28/2014	Vadim Nebogatov
vmn_upd20140329a.zip	214 KB	03/29/2014	Vadim Nebogatov
svl_upd20150205a.zip	26.6 KB	02/05/2015	Stanislav Lomany