# Database - Bug #6599

## minimal mode conversion problem with DMO name collision

07/14/2022 06:10 AM - Constantin Asofiei

| | | | |
|---|---|---|---|
| **Status:** | New | **Start date:** | |
| **Priority:** | Normal | **Due date:** | |
| **Assignee:** | Eric Faulhaber | **% Done:** | 0% |
| **Category:** | | **Estimated time:** | 0.00 hour |
| **Target version:** | | | |
| **billable:** | No | **case_num:** | |
| **vendor_id:** | GCD | **version:** | |

**Description**

### History

**#2 - 07/14/2022 06:16 AM - Constantin Asofiei**

After the 6129a rebase, a table named "property" no longer converts as property when conversion="minimal" is set at the schema namespace in p2j.cfg.xml.

The 4GL definition:

```
ADD TABLE "Property"
  AREA "Schema Area"
  DUMP-NAME "Property"

ADD FIELD "f1" OF "Property" AS character
  FORMAT "x(8)"
  INITIAL ""
  POSITION 2
  MAX-WIDTH 16
  ORDER 10

ADD FIELD "f2" of "Property" as character
  FORMAT "x(8)"
  INITIAL ""
  POSITION 3
  MAX-WIDTH 16
  ORDER 20

ADD INDEX "idx1" ON "Property"
  AREA "Schema Area"
  UNIQUE
  INDEX-FIELD "f2" ASCENDING
```

The DMO definition:

```
/**
 * Data Model Object corresponding with table {@code p2j_test.Property_} (previously {@code
 * p2j_test.Property}).
 */
@Table(name = "Property_", legacy = "Property")
@Indices(
{
   @Index(name = "Property__idx1", legacy = "idx1", unique = true, components =
   {
      @IndexComponent(name = "f2", legacy = "f2")
   })
})
public interface Property_
```

The postgresql DDL for the table:

```
create table Property_ (
   recid int8 not null,
   f1 text,
   f2 text,
   primary key (recid)
);
```

Previously the sql table name was property (without the trailing underscore).

I've noticed this because I imported the database in the wrong cluster, and the FWD server was using the old cluster where 'property' table exists.

**#3 - 07/27/2022 06:15 AM - Constantin Asofiei**

Eric, can this be fixed? Although is not a problem when running the converted code, this is a problem when executing direct SQLs via JDBC - as the SQL table name no longer matches the minimal conversion rules.

**#4 - 07/27/2022 03:43 PM - Eric Faulhaber**

Constantin, it looks like you made this change very early in 6129a (rev 13815), apparently to address cases of conflict between table and DMO annotation names (for instance, Property). This was long before the minimal mode changes went in (6129a/13860), so I'm not sure why this just showed up after a rebase.

Rev 6129a/13815 makes this change to NameConverter.java:

```
=== modified file 'src/com/goldencode/p2j/convert/NameConverter.java'
--- src/com/goldencode/p2j/convert/NameConverter.java    2022-02-10 13:49:14 +0000
+++ src/com/goldencode/p2j/convert/NameConverter.java    2022-04-26 13:04:51 +0000
@@ -125,6 +125,7 @@
 **     OM  20210121          Tests against reservedSQL in lowercase because SQL is case insensitive.
 **     OM  20210203          Avoid collision of buffer name and class with SQL keywords and java.lang classes
.
 **     CA  20220210          Added reservedFWD, which contains the BaseDataType types.
+**     CA  20220426          Added the FWD DMO annotations as reserved class names.
 */

 /*
@@ -188,6 +189,8 @@
 import com.goldencode.util.Stack;
 import com.goldencode.p2j.cfg.*;
 import com.goldencode.p2j.persist.*;
+import com.goldencode.p2j.persist.annotation.*;
+import com.goldencode.p2j.persist.annotation.Trigger;
 import com.goldencode.p2j.persist.dialect.*;
 import com.goldencode.p2j.persist.meta.*;
 import com.goldencode.p2j.schema.*;
@@ -429,6 +432,19 @@
          reservedFWD.add(type.getCls().getSimpleName());
       }
```

```
+       // add DMO annotations
+       reservedFWD.add(Index.class.getSimpleName());
+       reservedFWD.add(IndexComponent.class.getSimpleName());
+       reservedFWD.add(Indices.class.getSimpleName());
+       reservedFWD.add(Property.class.getSimpleName());
+       reservedFWD.add(Relation.class.getSimpleName());
+       reservedFWD.add(RelationComponent.class.getSimpleName());
+       reservedFWD.add(Relations.class.getSimpleName());
+       reservedFWD.add(Sequence.class.getSimpleName());
+       reservedFWD.add(Table.class.getSimpleName());
+       reservedFWD.add(Trigger.class.getSimpleName());
+       reservedFWD.add(Triggers.class.getSimpleName());
+
        // WARNING: Adding other classes/interfaces to this list may invalidated the already implemented
        // skeleton classes, as these must not collide with statically emitted APIs during conversion.
        Class<?>[] staticImportClasses =
```

The reservedFWD set is used in resolvePossibleKeywordConflict, perhaps too aggressively for SQL names in the case of some of these annotations. Some of them overlap with reserved SQL keywords, but others (including Property) do not.

However, even if we were to deal differently with the subset of words that are not SQL reserved keywords (e.g., IndexComponent, Property, Indices, etc.), we would still need to avoid conflicts with these annotation names on the Java side. So, with Property, for instance, we would still have a conflict with a DMO interface named Property, even if there was no longer a conflict with a table named property.

I'm not sure of the best way to resolve this, since the rev 13815 changes seems like it was made for a legitimate case of table names conflicting with DMO annotation names. We could potentially rename the DMO annotations to something less likely to collide with 4GL table names, but the potential for conflict, while smaller, would remain. This also would likely result in less meaningful annotation names.

Thoughts?

**#5 - 07/27/2022 04:02 PM - Constantin Asofiei**

Eric, minimal mode conversion was added after I my NameConverter.java change - and conversion was fine, until the rebase.

The issue was introduced after I rebased 6129a from 3821c, most likely in 3821c/14018, where a table with legacy name MyTable now converts to SQL table my_table instead of mytable (i.e. case is preserved to split it into words, previously was using the lower case name thus word splitting was lost).

I don't want to rename the annotations, this will just move the issue down the road.

I think the correct way is to not link the DMO name with the SQL table name, for minimal mode conversion - this is the only case AFAIK where we need legacy table names emitted as SQL table names.

**#6 - 08/01/2022 09:58 AM - Greg Shah**

*- Assignee set to Eric Faulhaber*

**#7 - 08/15/2022 11:34 AM - Eric Faulhaber**

Constantin Asofiei wrote:

> I think the correct way is to not link the DMO name with the SQL table name, for minimal mode conversion - this is the only case AFAIK where we need legacy table names emitted as SQL table names.

Yes, not sure why Java limitations are being applied to SQL names and vice versa. So, are you saying an acceptable outcome would be the following?

Given a legacy name property:

- DMO interface name: Property_
- SQL table name: property

**#8 - 08/15/2022 12:00 PM - Constantin Asofiei**

Eric Faulhaber wrote:

> Constantin Asofiei wrote:
>
> > I think the correct way is to not link the DMO name with the SQL table name, for minimal mode conversion - this is the only case AFAIK where we need legacy table names emitted as SQL table names.
>
> Yes, not sure why Java limitations are being applied to SQL names and vice versa. So, are you saying an acceptable outcome would be the following?
>
> Given a legacy name property:
>
> - DMO interface name: Property_
> - SQL table name: property

Yes, this works for me.  Your choice if this applies only to "minimal" mode or for any other mode (I don't think we can change the SQL name for existing applications moved away from 4GL).

**#9 - 08/15/2022 12:50 PM - Ovidiu Maxiniuc**

Constantin Asofiei wrote:

> Eric Faulhaber wrote:
>
> > Constantin Asofiei wrote:
> >
> > > I think the correct way is to not link the DMO name with the SQL table name, for minimal mode conversion - this is the only case AFAIK where we need legacy table names emitted as SQL table names.
> >
> > Yes, not sure why Java limitations are being applied to SQL names and vice versa. So, are you saying an acceptable outcome would be the following?
> >
> > Given a legacy name property:
> >
> > - DMO interface name: Property_
> > - SQL table name: property
>
> Yes, this works for me.  Your choice if this applies only to "minimal" mode or for any other mode (I don't think we can change the SQL name for existing applications moved away from 4GL).

The SQL column/table names must be checked against the getReservedKeywords() of the current dialect. But, if I figure it out correctly, there is at least a place where the property/table name is a merge between Java and SQL naming.
However there are two issues here:

- the column name in the Property annotation is the same for all dialects. This means it must be validated (excluded) against the list of reserved keywords of all known dialects. Or at least for the list of the dialects the conversion is ran;
- secondly, I did a quick tests with a table named Property having an homonym field. The result was an interface Property. This name is normally, fine, but we have the @Property annotation and the Java compiler complained about this and refused to compile the file. Anyway, this can be easily fixed by adding the @Property (and probably a good idea is to add all other persistence annotation classes) to reservedFWD set of NameConverter. DataModelObject, too.

**#10 - 08/15/2022 12:52 PM - Ovidiu Maxiniuc**

Sorry, I was focused on the very last entry here. I just saw the solution I proposed after clicking submit. However, DataModelObject seems to be missing.

**#11 - 08/15/2022 01:11 PM - Constantin Asofiei**

Ovidiu, this is related to 6129a branch, where DMO annotations are registered in NameConverter.