QGIS Application - Bug report #15188
Importing .shp files into PostGIS fails (Error 7)

2016-07-01 05:51 AM - R. R.

Status:	Closed		
Priority:	Low		
Assignee:	Sandro Santilli		
Category:	Data Provider/PostGIS		
Affected QGIS version:	2.16.3	**Regression?:**	No
Operating System:		**Easy fix?:**	No
Pull Request or Patch supplied:	No	**Resolution:**	
Crashes QGIS or corrupts data:	No	**Copied to github as #:**	23127

Description

Importing data into PostGIS by Drag&Drop or using the 'Import layer/file' icon fails for some .shp files (QGIS master commit:600ff4f):

> Error 7
> Feature write errors:
> Creation error for features from #0 to #0. Provider errors was:
> PostGIS error while adding features: FEHLER: Feldüberlauf bei Typ „numeric“
> DETAIL: Ein Feld mit Präzision 18, Skala 11 muss beim Runden einen Betrag von weniger als 10^7 ergeben.
>
> Only 0 of 2 features written.

In QGIS 2.14.3 everything works as expected.

Related issues:

Related to QGIS Application - Bug report # 11755: Real precision (Shapefile)	**Closed**	**2014-11-28**

Associated revisions

Revision a985d8c9 - 2016-10-12 12:58 PM - Sandro Santilli

Fix bogus precision/scale in PostgreSQL for double values

This reverts commit 92f71b696ca93c792ae5602ed82863fcef0e5006,
which broke import of legit shapefiles by assuming wrong
semantic for the non-constraining QgsField length/precision
attributes.

Closes #15188

Includes test

Revision d0b3430e - 2016-10-12 01:01 PM - Sandro Santilli

Fix bogus precision/scale in PostgreSQL for double values

This reverts commit 92f71b696ca93c792ae5602ed82863fcef0e5006,
which broke import of legit shapefiles by assuming wrong
semantic for the non-constraining QgsField length/precision
attributes.

Closes #15188

Includes test

Revision 0f4cba5c - 2016-10-12 08:48 PM - Sandro Santilli

Fix bogus precision/scale in PostgreSQL for double values

This reverts commit 92f71b696ca93c792ae5602ed82863fcef0e5006,
which broke import of legit shapefiles by assuming wrong
semantic for the non-constraining QgsField length/precision
attributes.

Closes #15188

Includes test

History

#1 - 2016-07-01 06:45 AM - Giovanni Manghi

- Status changed from Open to Feedback

please attach sample data. Thanks!

#2 - 2016-07-01 07:30 AM - R. R.

- File shp_test.zip added

#3 - 2016-07-03 12:23 AM - Giovanni Manghi

- Status changed from Feedback to Open

#4 - 2016-07-03 05:33 AM - R. R.

- File error7_qgis_2_15_600ff4f_ubuntu_16_04.png added

Also confirmed on Ubuntu (see screenshot).

#5 - 2016-07-05 09:32 AM - Martin Dobias

Hmm this is an interesting problem... It is all about width/precision issues of the SHAPE_Area field.

OGR reports that the field has width=19 and precision=11. In OGR the width includes also the decimal point. QGIS reports the field as with=18 and precision=11 which is correct (QGIS follows SQL convention of numeric(width,precision) data type where "width" does not include decimal point - only significant numbers). During export, QGIS therefore creates field numeric(18,11) for SHAPE_Area, which is correct.

Now here is the interesting bit: the stored value is "11456942.7758000010" that actually needs field type numeric(18,10) to be stored, the number is simply too big for numeric(18,11).

ogr2ogr works just fine because it handles the width in the same manner for shapefiles and postgres. Just tested and ogr2ogr indeed looses precision of last digits when converting from Postgres to shapefile.

One could argue that 1) the input shapefile data are badly written and 2) ogr2ogr looses precision, while QGIS handles everything correctly and the error message thrown during import to DB is perfectly valid.

It works in 2.14 without any errors because 2.14 does not preserve the exact numerical type and uses float8 instead (which looses some precision). Even though most of the time people probably do not care about that, in some cases such loss of precision may be an issue. OGR for example has an option for Postgres driver ("PRECISION") which allows users to say whether to use exact width/precision from the source field or whether to simply use float8.

By the way there is a related closed bug #11755

Opinions on what to do with it are most welcome :-)

#6 - 2016-07-05 09:38 AM - Martin Dobias

For the completeness, the change in behavior to not loose precision was introduced in this PR: https://github.com/qgis/QGIS/pull/3098

#7 - 2016-07-07 01:42 AM - Andreas Neumann

I will point this to Even to see what he thinks.

Would be nice if the user could still import it. QGIS could handle this difference of width/precision reporting between QGIS and OGR or OGR could be fixed. For QGIS 2.14/2.16, as OGR won't be fixed in the same time, it would be nice if QGIS could handle this for now, until it is potentially aligned in OGR (or not).

#8 - 2016-07-07 04:23 AM - Even Rouault

I pretty much agree with Martin's analysis :
- OGR field and width semantics likely originates from the shapefile/DBFfile and MapInfo convention, ie OGR width is the number of characters needed including leading minus sign and decimal point, and precision is the number of decimal figures
- The OGR PostgreSQL driver should likely do the adjustment to convert between the OGR width/precision and SQL precision/scale (https://www.postgresql.org/docs/9.0/static/datatype-numeric.html: beware of naming clashes ! OGR width ~= SQL precision and OGR precision = SQL scale !), but currently it does not.

There's not really a way of doing a perfect round-tripping because of the optional minus sign. So if you translate from SQL convention to OGR convention, you must compute OGR width as SQL precision + 1 (for minus sign) + 1 (for decimal point). But on the reverse way, for safety, you must do SQL precision = OGR width - 1, so if you do several roundtripping, the OGR width/SQL precision will keep growing. Apart from potentially doing the above fix in the OGR PG driver (which would break importing this shapefile because the DBF width.precision isn't consistant with the content), at this point of GDAL history, we wouldn't probably want to change the semantics of OGR width and precision in the abstract model as it could break user expectations.

#9 - 2016-07-09 02:09 PM - Jürgen Fischer

Andreas Neumann wrote:

> *I will point this to Even to see what he thinks.*
>
> *Would be nice if the user could still import it. QGIS could handle this difference of width/precision reporting between QGIS and OGR or OGR could be fixed. For QGIS 2.14/2.16, as OGR won't be fixed in the same time, it would be nice if QGIS could handle this for now, until it is potentially aligned in OGR (or not).*

That was what #11755 (commit:2fcbc8b) is about. The actual problem here is that the value in the DBF is too big for the definition of the DBF field (not sure how that works). I came to the same conclusion - but didn't have a good/quick solutions either (and moved on...)

#10 - 2016-09-14 04:43 AM - Willem Hoffmans

I've seen the same problem loading a single .dbf file into PostGIS via DB Manager.

An easy workaround is to use Shapefile & DBF Loader in PgAdmin to load the file directly into PostGIS, without use of QGIS. This worked well for me.

#11 - 2016-10-11 12:22 AM - Sandro Santilli

- Assignee set to Sandro Santilli
- Affected QGIS version changed from master to 2.16.3

Error confirmed with 2.16.3:

> Error 7
> Feature write errors:
> Creation error for features from #0 to #0. Provider errors was:
> PostGIS error while adding features: ERROR: numeric field overflow
> DETAIL: A field with precision 18, scale 11 must round to an absolute value less than 10^7.
>
> Only 0 of 1 features written.

2.14.7 works fine.

#12 - 2016-10-11 12:28 AM - Sandro Santilli

PostGIS loader (shp2pgsql) behaves as QGIS-2.14, sing "numeric" only when DBF width is > 18.
What was the rationale to change that behavior ? The referenced PR, which was merged as commit:92f71b696ca93c792ae5602ed82863fcef0e5006 in the 2.14 branch, did not contain an automated testcase.

#13 - 2016-10-11 12:34 AM - Sandro Santilli

Ok I was wrong, shp2pgsql (from PostGIS) behavior is to specify a width, but not a precision.
Doing so within QGIS results in a successful import. I'll look at automating a test for this.

#14 - 2016-10-11 12:34 AM - Sandro Santilli

- Status changed from Open to In Progress

#15 - 2016-10-11 12:34 AM - Sandro Santilli

- Category changed from DB Manager to Data Provider/PostGIS

#16 - 2016-10-11 12:45 AM - Sandro Santilli

Actually, I'm taking this back. shp2pgsql, which uses shapelib, reports the field to have width=19 and precision=11, but then doesn't set precision/width of the target field when the input is "FTDouble" (only does for input "integer").

So we're back to the question: why do we want to be strict about precision ?

#17 - 2016-10-11 01:33 AM - Sandro Santilli

So upon further digging, I've found that OGR correctly reports width as 19 (which would accept the value) but QGIS is reducing it by 1 in QgsOgrProvider::loadFields, for unknown reasons (there's no comment explaining that).

#18 - 2016-10-11 01:37 AM - Sandro Santilli

I confirm that removing the unit-subtraction from QgsOgrProvider::loadFields fixes the import.
The QgsField::length member is not documented either, so it isn't easy to know why would anyone change it to what value. Lack of documentation about it seems to be the main problem here.

#19 - 2016-10-11 02:10 AM - Sandro Santilli

I also confirm the other fix is to perform unit-addition in QgsPostgresProvider::convertField, but then the QgsField::length would be different if you load from shapefile or from the imported PostgreSQL table. Semantic of QgsField::length should be carefully defined, for the Double type.

Note that the PostgreSQL "numeric" type definition interprets the precision and scale modifiers as the total number of digits (not including the comma) and the number of digits on the right of the comma. Setting those values straight would require knowing in advance how many digits would possibly be on the left of the comma, and I don't think DBF would contain that info.

Are we sure we want to constraint ?

#20 - 2016-10-11 03:14 AM - Jürgen Fischer

Sandro Santilli wrote:

> *I also confirm the other fix is to perform unit-addition in QgsPostgresProvider::convertField, but then the QgsField::length would be different if you load from shapefile or from the imported PostgreSQL table. Semantic of QgsField::length should be carefully defined, for the Double type.*

It should follow what also the edit widgets expect (and was at some point).

#21 - 2016-10-11 03:23 AM - Sandro Santilli

I see by default there's always "text edit" as the edit widget associated with floating point fields, but I'm no expert there

#22 - 2016-10-11 05:49 AM - Jürgen Fischer

Sandro Santilli wrote:

> *I see by default there's always "text edit" as the edit widget associated with floating point fields, but I'm no expert there*

with a QgsFieldValidator that follows the field definition.

#23 - 2016-10-11 09:57 AM - Sandro Santilli

I've filed a PR with a testcase and a revert of the offending commit:
https://github.com/qgis/QGIS/pull/3590

The testcase checks for being able to import to PostgreSQL a memory layer in which a field has length/precision of 6/4 and a value of 123.456. The test fails before the revert. The test does not check the ability to retain (round-trip) the length/precision information, and I think it doesn't make sense to even try, given the inability of QgsField to actually use those parameters as constraining ones.

#24 - 2016-10-12 04:21 AM - Sandro Santilli

- Status changed from In Progress to Closed

Fixed in changeset commit:"a985d8c9f97554eaf84125bc6d458aa46088a0b5".

#25 - 2016-10-19 01:53 AM - Arnaud Morvan

Hello Sandro,

It's me that change this some times ago, because the french ministry of developement says that importing some data from postgis db to another postgis db create the new table with floats intead of decimal fields.

Note that using floats intead of decimals under 18 digits do not fix the problem over 18 digits.

#26 - 2016-10-19 02:03 AM - Sandro Santilli

Arnaud: note that as long as we store the value in a double, 18 digits just won't fit, so the value would be changed anyway.
PostgreSQL is able to store 18 digits, in a "numeric" (need not specify scale/precision for that), but qgis would still write an interpretation of the floating point number, with less precision, unless the value is kept as a string in qgis.

#27 - 2017-07-11 12:17 AM - Mike Taves

- Description updated
- Priority changed from Severe/Regression to Low

Sandro Santilli wrote:

> *So upon further digging, I've found that OGR correctly reports width as 19 (which would accept the value) but QGIS is reducing it by 1 in QgsOgrProvider::loadFields, for unknown reasons (there's no comment explaining that).*

This is the behavior of Esri products. On the same shapefile, a field in ArcCatalog will have (e.g.) Precision 23 and Scale 15. With OGR, it will have a width 24 and precision 15.

Files

error7.png	52.1 KB	2016-07-01	R. R.
shp_test.zip	1.72 KB	2016-07-01	R. R.
error7_qgis_2_15_600ff4f_ubuntu_16_04.png	87.5 KB	2016-07-03	R. R.