QGIS Application - Feature request #62
Qgis only supports postgres keys of type int

2006-04-06 12:06 AM - Gavin Macaulay -

Status:	Closed		
Priority:	Low		
Assignee:	Jürgen Fischer		
Category:	Data Provider		
Pull Request or Patch supplied:	No	**Resolution:**	fixed
Easy fix?:	No	**Copied to github as #:**	10121

Description

Transfer of item 1337919 from sourceforge.

Qgis should support postgres table keys of types other than int. There are workarounds for most situations but the better solution would be to directly support other key types.

History

#1 - 2006-04-07 07:00 AM - Steve Halasz -

A Debian was filed about this issue:

http://bugs.debian.org/355944

#2 - 2008-08-30 02:30 PM - gjm -

An email from the developers mailing list (Sun, 31 Aug 2008 02:42:42 +1200):

> Currently QGIS requires a int4 column with unique IDs for accessing data
> in a [[PostGIS]] database. I routinely work with [[OpenStreetMap]] data which
> uses int8 for its IDs. Currently the OSM IDs still fit into an int4, but
> this will not work forever. At the moment its a bit of a hassle that I
> always have to keep this limitation in mind.
>
> Is the limitation on int4 columns something hardcoded deep in the code
> or is it something that could be changed reasonably easy? I understand
> the need for a unique ID column, but it would be nice if the data type
> was more flexible. I think at least int2, int4, and int8 should be
> supported.
>
> Jochen Topf

#3 - 2008-08-30 07:05 PM - Jürgen Fischer

see also http://lists.osgeo.org/pipermail/qgis-developer/2008-August/004714.html

#4 - 2008-09-02 10:46 PM - springmeyer -

+1 to this being made more flexible. uDig/geotools provides a much more noticeably more flexible approach to reading in postgis tablea (even without a unique id field), so perhaps this could might be a good reference:

http://svn.geotools.org/trunk/modules/plugin/postgis/src/main/java/org/geotools/data/postgis/

autogeneration of fids:
http://svn.geotools.org/trunk/modules/plugin/postgis/src/main/java/org/geotools/data/postgis/fidmapper/

#5 - 2009-04-07 05:18 AM - Jürgen Fischer

since commit:f8277d91 (SVN r10475) the postgres providers resorts to using ctid as feature id, if no other usable primary key is found.

As a ctid currently consists of a 32bit block number and and 16bit offset, that option is limited to tables that have block numbers below 0x10000.

That problem would also disappear by applying the patch above to implement 64bit feature ids.

#6 - 2009-10-02 12:43 AM - Paolo Cavallini

What prevents us from applying this patch?

#7 - 2010-10-11 08:07 AM - willfurnass -

It would also be nice if [[PostGIS]] sequences could be used for layer keys.

I frequently wish to view the result of applying [[PostGIS]] aggregate functions using QGIS but this can't be done using relational views as the results of such queries lack anything that could be used as a primary key; I therefore end up littering my db with temporary tables created using 'SELECT...INTO...' so as to view the results of aggregate queries in QGIS. Allowing a self-incrementing sequence col within a view to be used as a layer key would be a (slightly ugly) way around this.

#8 - 2010-10-11 03:48 PM - Jürgen Fischer

Replying to [comment:17 willfurnass]:

> *It would also be nice if [[PostGIS]] sequences could be used for layer keys.*
> *I frequently wish to view the result of applying [[PostGIS]] aggregate functions using QGIS but this can't be done using relational views as the results of such queries lack anything that could be used as a primary key; I therefore end up littering my db with temporary tables created using 'SELECT...INTO...' so as to view the results of aggregate queries in QGIS. Allowing a self-incrementing sequence col within a view to be used as a layer key would be a (slightly ugly) way around this.*

The feature ids are necessary to identify which feature is to be deleted or updated. The table/view is not only queried once, but eg. for each render operation with the current extent in the where clause. And in every query each feature is supposed to have the same id as in the previous queries. I'm not sure how that should work with sequences in views.

#9 - 2010-10-14 03:20 AM - willfurnass -

> *The feature ids are necessary to identify which feature is to be deleted or updated. The table/view is not only queried once, but eg. for each render operation with the current extent in the where clause. And in every query each feature is supposed to have the same id as in the previous queries. I'm not sure how that should work with sequences in views.*

Interestingly both of the following views can be loaded by uDig.

CREATE VIEW myview AS (SELECT zone_ref, ST_ConvexHull(ST_Collect(wkb_geometry)) FROM table_of_points GROUP BY zone_ref);

CREATE VIEW myview_with_id AS (SELECT nextval('some_seq'), zone_ref, ST_ConvexHull(ST_Collect(wkb_geometry)) FROM table_of_points GROUP BY zone_ref);

Does anyone know how uDig internally handles the rendering and manipulation of tables/views lacking a UNIQUE constraint? Would it be possible to allow for the viewing of such relations in some sort of read-only mode? Apologies if I'm missing something obvious here.

#10 - 2011-12-16 01:59 PM - Giovanni Manghi

- Target version changed from Version 1.7.0 to Version 1.7.4

#11 - 2011-12-17 01:48 AM - Jürgen Fischer

- Resolution set to fixed

- Pull Request or Patch supplied set to No

64 bit feature id support patch applied in commit:5a3a87fde2f0644fb6e89a1ff1e5af60f03ea6b2 (back in July)

For the view key issue see also http://linfiniti.com/2011/11/adding-a-counter-to-postgresql-query-results/

#12 - 2011-12-17 01:49 AM - Jürgen Fischer

- Status changed from Open to Closed

Files

featureid.diff	52.8 KB	2008-08-30	Jürgen Fischer