QGIS Application - Feature request #972
do locking when features are changed in PostGIS tables

2008-03-05 04:13 PM - Maciej Sieczka -

Status:	Open		
Priority:	Low		
Assignee:			
Category:	Data Provider		
Pull Request or Patch supplied:	No	**Resolution:**	
Easy fix?:	No	**Copied to github as #:**	11031

Description

QGIS doesn't prevent multiple users from editing the same table at the same time, which sooner or later results in overlapping features and other surprises that ruin the party. I asked on Postgres ML whether [[PostGIS]] client software should do locking while table is being edited r1, and their answer is yes:

The application should either:

1. Take an advisory lock (see the functions/admin functions chapter) so that it can use another table to indicate which parts of the GIS are in use.

2. Check to see if the data changed while the user was editing but before committing (known as "optimistic locking"). Then give the user the option to overwrite/rollback.

A last resort would be locking rows or the whole table, since a user might click "edit" then go to lunch.

Certainly doing nothing isn't much use if you have multiple users editing.

Please fix QGIS in this regard. There's been a discussion about this on QGIS dev ML r2.

r1http://archives.postgresql.org/pgsql-general/2008-02/msg01285.php
r2http://www.nabble.com/forum/ViewPost.jtp?post=15721104&framed=y

Related issues:

Related to QGIS Application - Feature request # 15208: Add conflict managemen...	**Feedback**	**2016-07-04**

History

#1 - 2008-04-11 06:31 AM - Maciej Sieczka -

Replying to [comment:3 jef]:

Hi. Why do you think this is a minor issue? An application which allows for concurrent table use, should perform the necessary checks, in order to avoid data corruption, as a must, IMO.

#2 - 2008-04-11 08:20 AM - Jürgen Fischer

Replying to [comment:4 msieczka]:

> *Replying to [comment:3 jef]:*
>
> *Hi. Why do you think this is a minor issue? An application which allows for concurrent table use, should perform the necessary checks, in order to*

> *avoid data corruption, as a must, IMO.*

I see it as enhancement, not as critical bug. There's no crash and no data corruption. For now who saves last wins. But you're right, that applies to Type ant not priority - both adjusted.
BTW locking wouldn't address adding features, so without other means of coordination people might add the "same" features multiple times without noticing in the process, even if the cannot edit or delete existing features.

#3 - 2008-04-11 08:48 AM - Maciej Sieczka -

Replying to [comment:5 jef]:

> *I see it as enhancement, not as critical bug. There's no crash and no data corruption. For now who saves last wins.*

My point is that it never should have been implemented this way. I don't think that the fact it was designed wrong is an excuse for the data corruption it leads to.

> *But you're right, that applies to Type ant not priority - both adjusted.*
> *BTW locking wouldn't address adding features, so without other means of coordination people might add the "same" features multiple times without noticing in the process, even if the cannot edit or delete existing features.*

I'm affraid I don't follow. You mean that locking the table would not prevent adding features by others?

#4 - 2008-04-11 08:58 AM - Jürgen Fischer

Replying to [comment:6 msieczka]:

> *My point is that it never should have been implemented this way. I don't think that the fact it was designed wrong is an excuse for the data corruption it leads to.*

What data corruption?

>> *BTW locking wouldn't address adding features, so without other means of coordination people might add the "same" features multiple times without noticing in the process, even if the cannot edit or delete existing features.*

> *I'm affraid I don't follow. You mean that locking the table would not prevent adding features by others?*

The current approach described in r2 above, would be locking features not tables.
So once someone starts to modify or deletes a feature it is locked until commit, so nobody else can modify or delete concurrently. This would affect adding features at all.

#5 - 2008-04-11 08:59 AM - Jürgen Fischer

> *This would affect adding features at all.*

wouldn't that is.

#6 - 2008-04-12 05:52 AM - Maciej Sieczka -

Replying to [comment:7 jef]:

> *What data corruption?*

Eg. a polygon overlapping another one - each digitised by a different user, at the same time (ie. topology broken). Locking table write access for the user until he commits his changes would prevent such cases.

> *The current approach described in r2 above, would be locking features not tables.*

Locking on rows is often not enough for [[PostGIS]] tables IMO. Single rows (features) of a [[PostGIS]] table are **spatially** inderdependent, unlike in case of non-spatial data. Locking per feature does not improve the situation much. With per-row locking you can still have many users digitising (ie. editing the table) at the same time, which will lead to a mess in the data eventually.

> *So once someone starts to modify or deletes a feature it is locked until commit, so nobody else*
> *can modify or delete concurrently. This would affect adding features at all.*

I agree that locking per feature would be usefull in some setups. But I also believe that an exclusive lock on a whole table would be a safest approach. Probably it would be best to have this configurable in QGIS (as a global setting and a per-project override), so that the user could choose an optimal locking mechanism in his [[PostGIS]] usage scanario.

#7 - 2008-04-12 09:28 AM - Jürgen Fischer

Replying to [comment:9 msieczka]:

> *Replying to [comment:7 jef]:*
>
> > *What data corruption?*

> *Eg. a polygon overlapping another one - each digitised by a different user, at the same time (ie. topology broken). Locking table write access for the user until he*

commits his changes would prevent such cases.

I see. Table locking has something to it in that case.   What I don't like that it blocks users from editing features in different spatial areas.

But I wouldn't call that data corruption in our (trac) sense as it isn't qgis that corrupts the data, the user does. This could happen even without multiple users - give or take locking.

There should be a constraint on the database that prevents things like this from happening. In that case the commit of the changes would fail and you could align the geometry until the constraints are met and the commit succeeds. Those constraints would also apply to added features.

Without such constraints you would have to deal with the possibility of inconsistent topology anyway.

But I have to admit that I never have used topology constraints - if they exists at all in [[PostGIS]], I imagine that they probably have a big performance hit to them.

> *Single rows (features) of a [[PostGIS]] table are **spatially** inderdependent, unlike in case of non-spatial data.*

I tend to disagree. That also applies to non-spatial data, actually I think that that's the whole point of RDBMS with referential integrity and normalization etc.

BTW speaking of normalization a line topology would work in your case with row locking. Ok, probably not feasible as you loose the spot for the attributes that you used to assign to the polygons.

> *Probably it would be best to have this configurable in QGIS (as a global setting and a per-project override), so that the user could choose an optimal locking mechanism in his [[PostGIS]] usage scanario.*

Ok.

#8 - 2008-07-14 03:58 PM - Tim Sutton

A note on the meaning of 'major: does not work as expected'. This assignment should be used when a feature has been implemented (e.g. digitise a line) but some issue prevents the feature working. Minor priority should be used where by design or ommission a feature has not been implemented yet. In the case of this ticket I would contend the priority to be 'minor'. I'll leave the priority as is and let the chips fall where they may in this case....

#9 - 2011-07-03 05:14 PM - Nathan Woodrow

IMO locking database tables never makes any sense. One of the main reason to move to a database system rather then flat files is the ability for multiple people to edit the same layer at the same time.

Locking the row that is being edited and handling conflicts is a much better approach. MapInfo handles this by showing the user a dialog that says "This is on the server" and "Here are you local changes" What do you want me to keep? I'm not sure if QGIS has this already implemented but it might be something to consider.

#10 - 2012-10-06 02:34 AM - Pirmin Kalberer
- Target version changed from Version 2.0.0 to Future Release - Nice to have

#11 - 2014-06-24 09:40 AM - Giovanni Manghi
- Assignee deleted (Jürgen Fischer)
- Operating System deleted (All)
- Status info deleted (0)
- Pull Request or Patch supplied set to No

> *Locking the row that is being edited and handling conflicts is a much better approach. MapInfo handles this by showing the user a dialog that says "This is on the server" and "Here are you local changes" What do you want me to keep? I'm not sure if QGIS has this already implemented but it might be something to consider.*

this would be a really nice feature to have (locking row and handling conflicts). It is also an increasingly asked feature because QGIS and PostGIS are more and more popular...

#12 - 2014-06-24 10:29 AM - Antonio Locandro

I agree with Nathan that is one of the reasons to move to a database system. Maybe implementing something similar to what MapInfo does would be better than the horrible other COTS GIS software lock all database/tables for edits

#13 - 2014-06-25 01:08 AM - Matthias Kuhn

I would prefer to go for proper transaction support and MVCC

#14 - 2017-05-01 12:50 AM - Giovanni Manghi
- Easy fix? set to No