QGIS Application - Bug report #9509
Discarding edits freezes qgis for a long time

2014-02-07 11:44 AM - Giovanni Manghi

Status:	Closed		
Priority:	Severe/Regression		
Assignee:			
Category:	Vectors		
Affected QGIS version:	master	**Regression?:**	No
Operating System:		**Easy fix?:**	No
Pull Request or Patch supplied:	No	**Resolution:**	
Crashes QGIS or corrupts data:	No	**Copied to github as #:**	18100

Description

This is a follow up of #9315

Open a fair large vector like http://idena.navarra.es/descargas/CARTO1_Lin_6CNivelD.zip

with the field calculator update a column with whatever you want, then toggle editing and discard changes: QGIS freezes for a looooong time.

On the other hand with the provided dataset if you choose to save edits it takes just a few seconds to save.

Related issues:

Related to QGIS Application - Bug report # 9315: Field Calculator, slow perfo...	**Closed**	**2014-01-09**
Duplicated by QGIS Application - Bug report # 9519: Rollback operations slow ...	**Closed**	**2014-02-09**

Associated revisions

Revision 52616b65 - 2014-02-08 11:28 PM - Jürgen Fischer

vector layer: save old attribute values where available to speedup undo (and rollback) (fixes #9509)
field calculator & app: use wait cursor while calculating and committing or rolling back

Revision 5bee1721 - 2014-02-12 02:10 PM - Jürgen Fischer

- speed up undo of attribute added with field calculator (fixes #9509)
- also improves performance of FilterFid requests in edit mode

History

#1 - 2014-02-08 02:30 PM - Jürgen Fischer

- Status changed from Open to Closed

Fixed in changeset commit:"52616b6549bf43eee3cccc79f73aa890da5f3fab".

#2 - 2014-02-09 06:43 AM - Salvatore Larosa

For me it still not fixed. I am getting always a QGIS freezing on rollback action.

#3 - 2014-02-09 07:03 AM - Jürgen Fischer

Salvatore Larosa wrote:

> *For me it still not fixed. I am getting always a QGIS freezing on rollback action.*

It's faster. But the changes need to be rolled back one by one and therefore the time it takes to rollback is somewhat the same as to do the changes in the first place.

#4 - 2014-02-09 09:19 AM - Matthias Kuhn

- Status changed from Closed to Reopened

It's not quite the same, as when doing changes the first time via the field calculator, one iterator is opened and looped while in a rollback, there is one iterator per feature.

Quote from #9519

> *As far as I can tell, there are feature requests made for every single feature which was updated before (so all in the case mentioned above).*
> *I could imagine a possible approach would be to batch undo the affected rows and perform a single request with setFilterFids() to still only request the required rows, but with less roundtrips.*

#5 - 2014-02-09 09:25 AM - Jürgen Fischer

Matthias Kuhn wrote:

> *It's not quite the same, as when doing changes the first time via the field calculator, one iterator is opened and looped while in a rollback, there is one iterator per feature.*

Not anymore. That was the case when the previous values were not stored in the first undo command - therefore the original value had to be retrieved from the provider using a iterator. Now that only happens if an attribute change happens when the previous value isn't already available (which is not the case in the field calculator).

Still all subscribers to the attributeValueChanged signal must be notified that the value was changed back (BTW the dualview form view doesn't update in that case).

#6 - 2014-02-10 02:01 AM - Matthias Kuhn

Jürgen Fischer wrote:

> *Not anymore. That was the case when the previous values were not stored in the first undo command - therefore the original value had to be retrieved from the provider using a iterator. Now that only happens if an attribute change happens when the previous value isn't already available (which is not the case in the field calculator).*

So the problem is the field calculator not saving the previous values?

> *Still all subscribers to the attributeValueChanged signal must be notified that the value was changed back (BTW the dualview form view doesn't update in that case).*

I can take care of that. I think introducing an afterRollback signal should do the trick.

#7 - 2014-02-10 02:05 AM - Matthias Kuhn

Ah, the form view and not the table view.

/me should read more carefully.

I will take care of this for 2.4 with the new editor components.

#8 - 2014-02-10 02:18 AM - Jürgen Fischer

Matthias Kuhn wrote:

> *Jürgen Fischer wrote:*
>
> > *Not anymore. That was the case when the previous values were not stored in the first undo command - therefore the original value had to be retrieved from the provider using a iterator. Now that only happens if an attribute change happens when the previous value isn't already available (which is not the case in the field calculator).*
>
> *So the problem is the field calculator not saving the previous values?*

IMHO there is no problem anymore. The rollback is faster now, as it doesn't need to reload the values from the provider, but it still needs time because it needs to signal each undo - and that's by design.

We only have the attributeValueChange signal and subscribers can currently assume that they will be notified for each change. We could introduce a bulkAttributeValueChange signal (and undo command), that tells subscribers that all the values of an attribute changed (maybe include fid ranges or fid sets) and define that attributeValueChange doesn't cover bulk changes. But that would be a feature.

#9 - 2014-02-10 02:37 AM - Matthias Kuhn

I am not at home to test now but when I tested yesterday it seemed to me that the field calculator is still faster than the rollback - and they should emit the same amount of signals. Maybe the decrease in performance is related to debug output the rollback produces.

bulkAttributeValueChange signal would be fine, but on the other hand I experienced that caching values retrieved from attributeChanged signals and postponing heavy operations to editCommandEnded() works very well. So the signals themselves seem to be a minor problem as long as their implementations are not performance-killers. The only remaining problem with this approach I can think of is that if several consumers cache the values it's a waste of memory.

#10 - 2014-02-10 03:05 AM - Jürgen Fischer

Matthias Kuhn wrote:

> *I am not at home to test now but when I tested yesterday it seemed to me that the field calculator is still faster than the rollback - and they should emit the same amount of signals. Maybe the decrease in performance is related to debug output the rollback produces.*

Before or after commit:52616b6? I didn't do benchmarking either, but the test expression to update and undo I used was trivial.

#11 - 2014-02-11 10:15 PM - Matthias Kuhn

Tested again with current master.

I noticed, that when *updating* a column, the rollback is fast. But when I *add* a newly calculated column that's what takes so long to rollback.

In the latter case the debug output I mentioned is:

```
src/providers/postgres/qgspostgresconn.cpp: 825: (openCursor) Starting read-only transaction
src/providers/postgres/qgspostgresconn.cpp: 839: (closeCursor) Committing read-only transaction
src/providers/postgres/qgspostgresconn.cpp: 825: (openCursor) Starting read-only transaction
src/providers/postgres/qgspostgresconn.cpp: 839: (closeCursor) Committing read-only transaction
src/providers/postgres/qgspostgresconn.cpp: 825: (openCursor) Starting read-only transaction
src/providers/postgres/qgspostgresconn.cpp: 839: (closeCursor) Committing read-only transaction
src/providers/postgres/qgspostgresconn.cpp: 825: (openCursor) Starting read-only transaction
src/providers/postgres/qgspostgresconn.cpp: 839: (closeCursor) Committing read-only transaction
src/providers/postgres/qgspostgresconn.cpp: 825: (openCursor) Starting read-only transaction
src/providers/postgres/qgspostgresconn.cpp: 839: (closeCursor) Committing read-only transaction
src/providers/postgres/qgspostgresconn.cpp: 825: (openCursor) Starting read-only transaction
```

#12 - 2014-02-12 05:10 AM - Jürgen Fischer
- Status changed from Reopened to Closed

Fixed in changeset commit:"5bee17218db3576a6a172eec5ef6d555fc023b4d".