

---

# qgis-versioning Documentation

*Release latest*

February 18, 2016



<b>1</b>	<b>Introductory text</b>	<b>3</b>
<b>2</b>	<b>Requirements</b>	<b>5</b>
<b>3</b>	<b>Plugin functionality</b>	<b>7</b>
<b>4</b>	<b>Selecting features before checkout</b>	<b>17</b>
<b>5</b>	<b>Project contributors</b>	<b>19</b>



This is the temporary location of the `qgis-versioning plugin` documentation. Click this link for the [original documentation](#).

---

**Date** Last generated on 18-02-2016 at 09:34

---



---

# Introductory text

---

## 1.1 Summary

The `qgis-versioning` plugin provides versioning of geographical features stored in PostGIS databases. A typical use case met by the plugin involves one or more users checking out a local working copy to edit features in (potentially) offline mode using SpatiaLite and committing changes back to the PostGIS server.

More information can be found in the plugin's [original documentation](#) page.





---

## Requirements

---

This section is a list of both hard and soft requirements (aka suggestions).

### 2.1 Hard requirements

Hard requirements you can't really live without. The plugin may work but you may have real problems to make it work. The two basic hard requirements are about QGIS and PostgreSQL/PostGIS.

#### 2.1.1 QGIS

Recent versions of the plugin were tested with QGIS 2.8. It is worth mentioning that plugin versions < 0.2 were developed for older versions of QGIS, which may incidentally ship with older versions of spatialite/SQLite. As of version 0.2 (Aug 2015), [spatialite](#) version 4.x is supported by the plugin. This in turn makes any QGIS versions that come with older spatialite versions unusable with the plugin (for SL checkouts at least).

Another key dependency of the plugin is [ogr2ogr](#). Although the plugin does not depend on the most recent features of ogr2ogr, it is wise to stick to the version bundled in QGIS 2.8+ (plus because newer versions should be backwards compatible).

#### 2.1.2 PostgreSQL/PostGIS

The plugin should work on any minor revision of the PostgreSQL 9.x series. It was tested successfully on PostgreSQL 9.2 and 9.4.

There are no hard requirements on PostGIS as such on the part of the plugin. Packing the most recent version supported in your PostgreSQL installation should be sufficient.

#### 2.1.3 Naming conventions

Operation of the plugin is best ensured by sticking to the PostgreSQL naming rules. As suggested [here](#) :

PostgreSQL uses a single data type to define all object names: the name type. A value of type name is a string of 63 or fewer characters. A name must start with a letter or an underscore; the rest of the string can contain letters, digits, and underscores.

<b>Warning:</b> Do NOT use empty spaces in any identifier the plugin asks you to supply
---

Even though PostgreSQL object names can contain capital letters, the plugin does not currently support object names other than in lowercase letters (plus digits and underscores as mentioned above). Even though the plugin ensures some level of protection in that respect, it is best to stick to those conventions when naming a new PG checkout (see later for an explanation), a branch or any other name the plugin asks you to provide.

The same applies to spatialite filenames (SL checkout).

## 2.2 Soft requirements

Soft requirements are more like “best practice” suggestions. As the saying goes : Your Mileage May Vary.

### 2.2.1 Separate schema

As will be explained in more detail later in this document, the `qgis-versioning` plugin operates “historization” by adding columns to each table in a particular database together with a *revisions* table that holds all revision information. For a number of reasons, it is wise to isolate your geographic data in a schema **other** than the *public* schema.

As mentioned [here](#) :

”... store no data in the ‘public’ schema.”

The specific context of the previous quote pertains to backup and restore procedures but the advice also applies for the `qgis-versioning` plugin.









---

## Plugin functionality

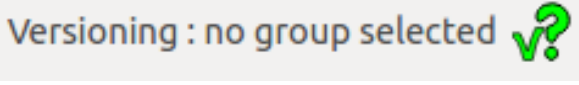
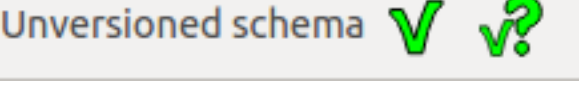
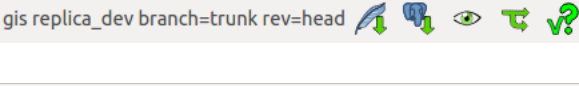
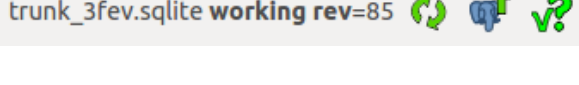
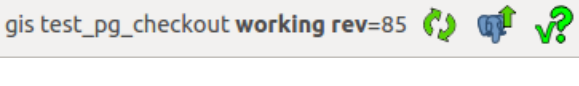
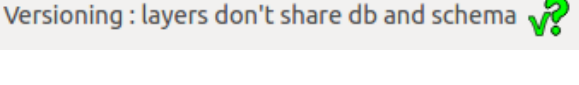
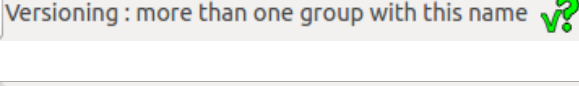
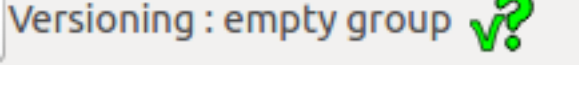
---

### 3.1 Summary

Depending on the type of layer group (unversioned/versioned PostGIS database, PostGIS/Spatialite working copy) the `qgis-versioning` plugin provides a different set of functionalities, as summarized in the following table.

Icon	Unversioned	Versioned	Working copy (PG/SL)	Definition
		X		Branching
		X		Checkout Spatialite
		X		Checkout PostGIS
			X	Commit changes
	X	X	X	Help
	X			Start versioning
			X	Check if working copy up to date
		X		View revisions

The following table shows the combination of plugin messages and icons as a function of group type selected in the QGIS legend.

Group type	Menu	Comments
No group		No group item is selected in the legend
Unversioned		Only option is to historize (green V)
Versioned		Textinfo (left) : DB schema branch= rev=
Working Copy (SL)		Textinfo (left) : filename working rev=
Working copy (PG)		Textinfo (left) : DB schema working rev=
Mixed layers		Layers in group do not share the same database or schema
Same Name		Groups must have different names
Empty group		Selected group is empty

## 3.2 Typical workflow

The following sections present how the `qgis-versioning` plugin is used in QGIS. Another section will detail what the plugin does in the database.

### 3.2.1 Unversioned database

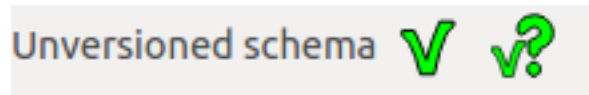
The sequence of operations begins with data in a PostGIS (PG) database that is loaded in a QGIS layer group.


---

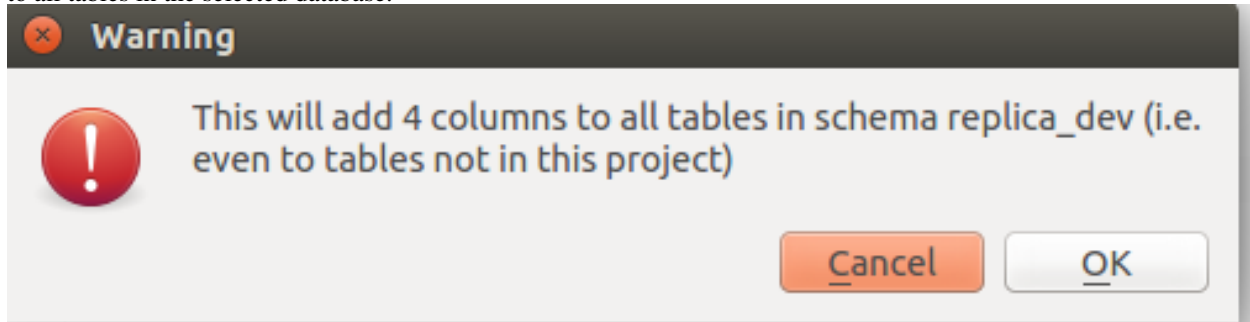
**Note:** The `qgis-versioning` plugin operates on QGIS layer groups, not on individual layers.

---

At that stage the data is unversioned and the only option on the layer group, except for help, is to “historize” the database.



Clicking on the historize button (  ) will generate a warning from the plugin that four new columns will be added to all tables in the selected database.

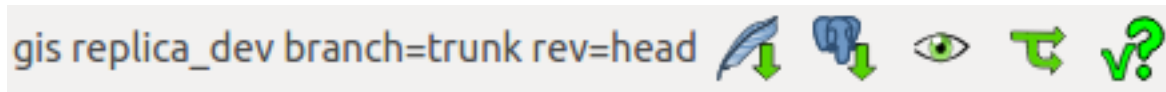


As the warning mentions, *all* datasets in the database will be versioned even though a subset of datasets (tables) was initially selected by the user to be loaded as layers in QGIS.

Upon accepting, the plugin creates a *versions* table in the formerly unversioned database schema, creates a new schema by appending “\_trunk\_rev\_head” to the current schema name, indicating the original schema now has one line of versioning called “trunk”, and loads the selected layers in a new QGIS group. That is the beginning of the versioning journey.

### 3.2.2 Versioned database

The plugin menu for a versioned layer group shows 5 icons.



On the left, the name of the database schema is shown. More specifically, the space-separated text items on the left identify four components :

- name of database
- name of schema
- name of branch (initial branch has default name *trunk*)
- revision number

Three operations can be performed on a versioned layer group :

1. branching
2. checking out a working copy
3. viewing specific revisions

#### Branching



Branching involves the creation of a new schema in the database. The new schema becomes another line of versioning of the original schema.

---

**Note:** Even though branches are to hold independent versioning histories, they still result in a “commit” in the *revisions* table.

---


## Checking out a working copy

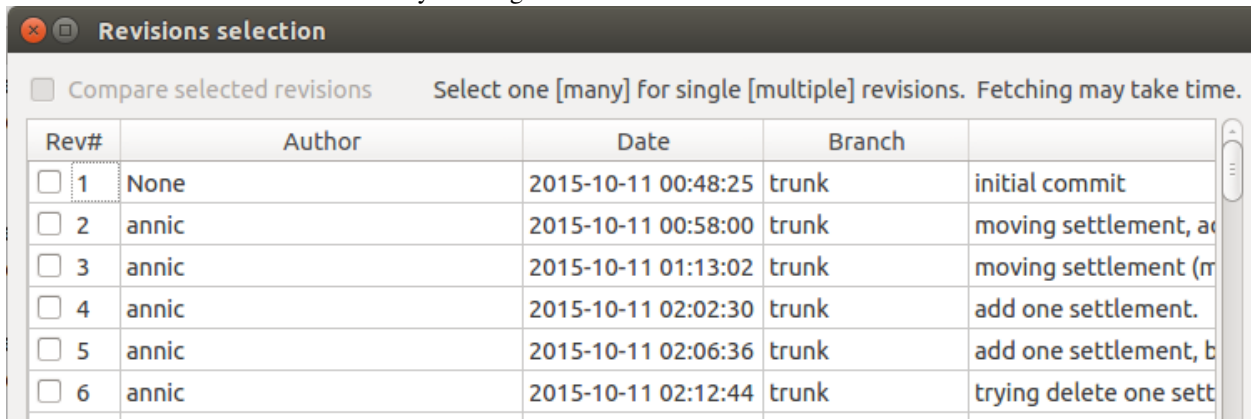
Checking out a working copy creates one of two new layer groups, either a PG checkout (  ) or a spatialite checkout (  ):

- a spatialite layer group, called *working copy* (or named with the full path name of the spatialite file created if a group called “working copy” already exists) is created in the legend
- a PG layer group, the name of which is made up of the user provided schema name

In both cases, properties of individual layers in the groups will clearly show provenance (spatialite filename on the file system for spatialite checkouts or “schema”.“view\_name” for PG checkouts).

## Viewing revisions

The view icon (  ) shows the contents of the *revisions* table stored in the schema that was originally versioned. The user can select one or more revisions by clicking the checkbox before the revision number.

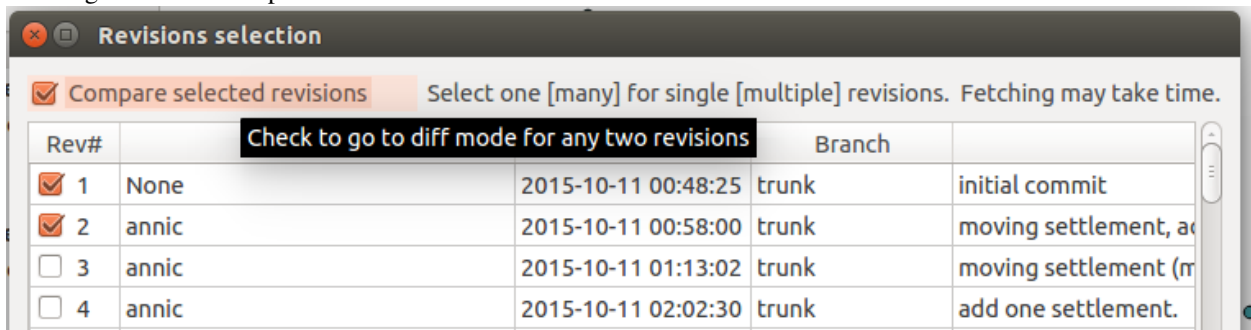


☐ Compare selected revisions      Select one [many] for single [multiple] revisions. Fetching may take time.

Rev#	Author	Date	Branch	
<input type="checkbox"/> 1	None	2015-10-11 00:48:25	trunk	initial commit
<input type="checkbox"/> 2	annic	2015-10-11 00:58:00	trunk	moving settlement, ac
<input type="checkbox"/> 3	annic	2015-10-11 01:13:02	trunk	moving settlement (m
<input type="checkbox"/> 4	annic	2015-10-11 02:02:30	trunk	add one settlement.
<input type="checkbox"/> 5	annic	2015-10-11 02:06:36	trunk	add one settlement, b
<input type="checkbox"/> 6	annic	2015-10-11 02:12:44	trunk	trying delete one sett

Selecting one or more revision numbers will result in one group per revision created in the QGIS legend tree. Each of those groups show all layers at the specific revision number. The default name of those groups is “branch\_name” + “revision” + the revision number, for example “trunk revision 1”.

At the top of the dialog, a checkbox called “Compare selected revisions” gets enabled when two revisions are checked, allowing the user to compare between the two revisions.

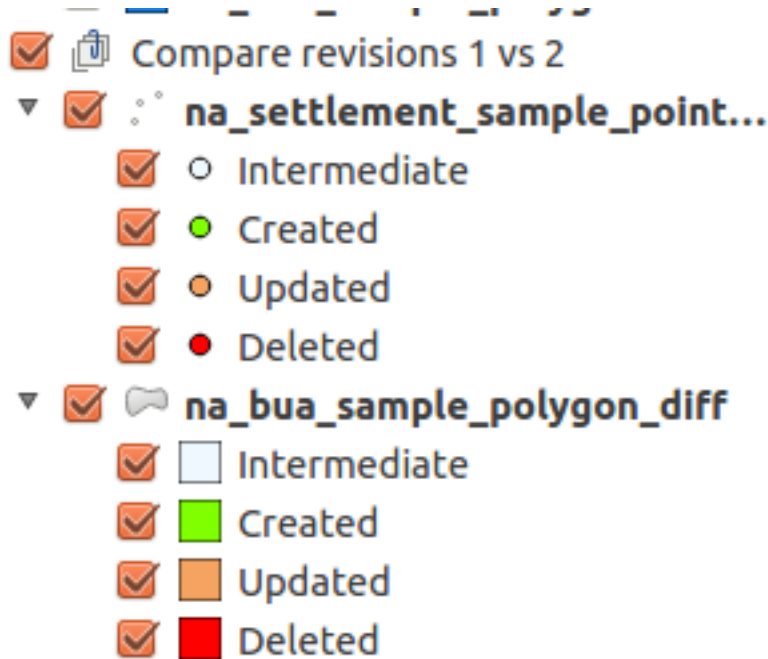


☒ Compare selected revisions      Select one [many] for single [multiple] revisions. Fetching may take time.

Check to go to diff mode for any two revisions

Rev#	Author	Date	Branch	
<input checked="" type="checkbox"/> 1	None	2015-10-11 00:48:25	trunk	initial commit
<input checked="" type="checkbox"/> 2	annic	2015-10-11 00:58:00	trunk	moving settlement, ac
<input type="checkbox"/> 3	annic	2015-10-11 01:13:02	trunk	moving settlement (m
<input type="checkbox"/> 4	annic	2015-10-11 02:02:30	trunk	add one settlement.

Instead of getting two layer groups each containing all features, clicking on that checkbox creates a single group called “Compare revisions X vs Y” (with  $X < Y$ ) where features that differ between the two compared revisions are given a rule-based symbology to highlight features that were created, updated or deleted.



A special case named “intermediate” identifies features that are “transient” items. An example would be a point feature that would have been moved between the two revisions but which has a parent feature in revision X and a child feature in revision Y.

**Note:** The “Compare selected revisions” checkbox is automatically unchecked and disabled if the number of selected revisions is not equal to two (2).

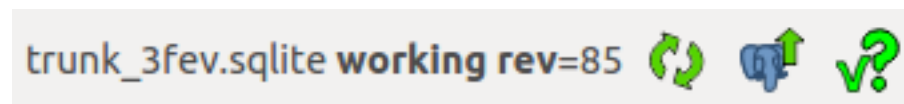
### 3.2.3 Working copy

The `qgis-versioning` plugin allows users to work on two types of working copies : PostGIS and Spatialite. Once a database is versioned, users can *checkout* a working copy in either PostGIS or Spatialite formats. In the former case, a new schema is created on the PostGIS server and a copy of the features for the selected layers is created. In the latter, a local Spatialite file is created.

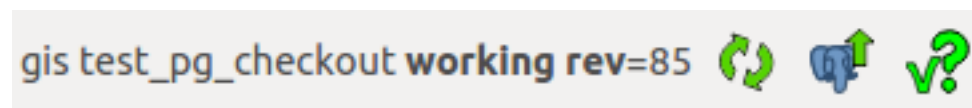
A peculiarity of spatialite checkouts is that features can be selected prior to checking out as explained in [Selecting features before checkout](#). Working copies can also be updated with changes committed by other users in the central PG database.



The following image shows the three icons found in the menu bar for a working copy, in this case of :

- a spatialite file



- PostGIS



On the left, either the name of the schema in the central database(for PG checkout) or the name of the spatialite file appears together with the current revision number. The two basic operations that can be performed on a working copy are to either update (  ) or to commit changes (  ).

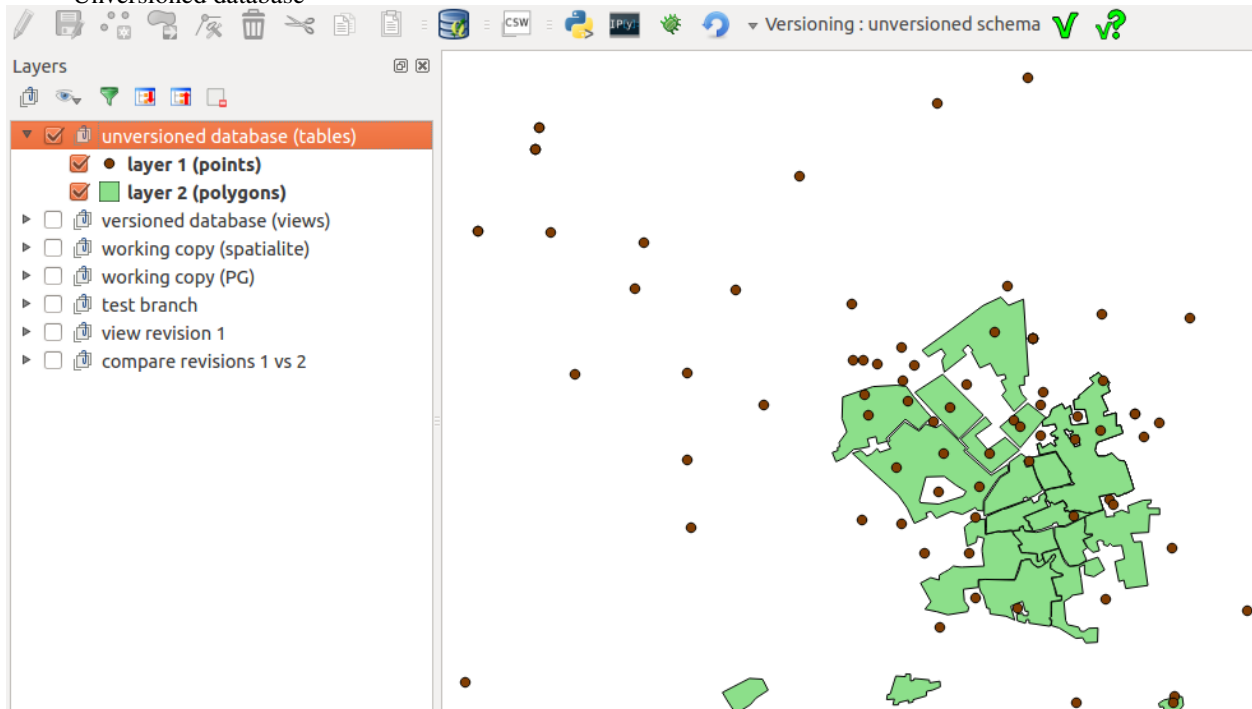
Updating implies synchronizing the current working copy with the central database. If the current working copy is behind the central database data is uploaded to the working copy. Data is either integrated directly in the working copy or a conflict resolution workflow is launched in the event local edits conflict with database revisions newer than that at which the working copy was checked out.

Committing changes is rather self explanatory. After changes were made in the working copy, they can be committed to the central database where they will be integrated and the revisions table updated with a new rev count.

### 3.3 Plugin group types in pictures

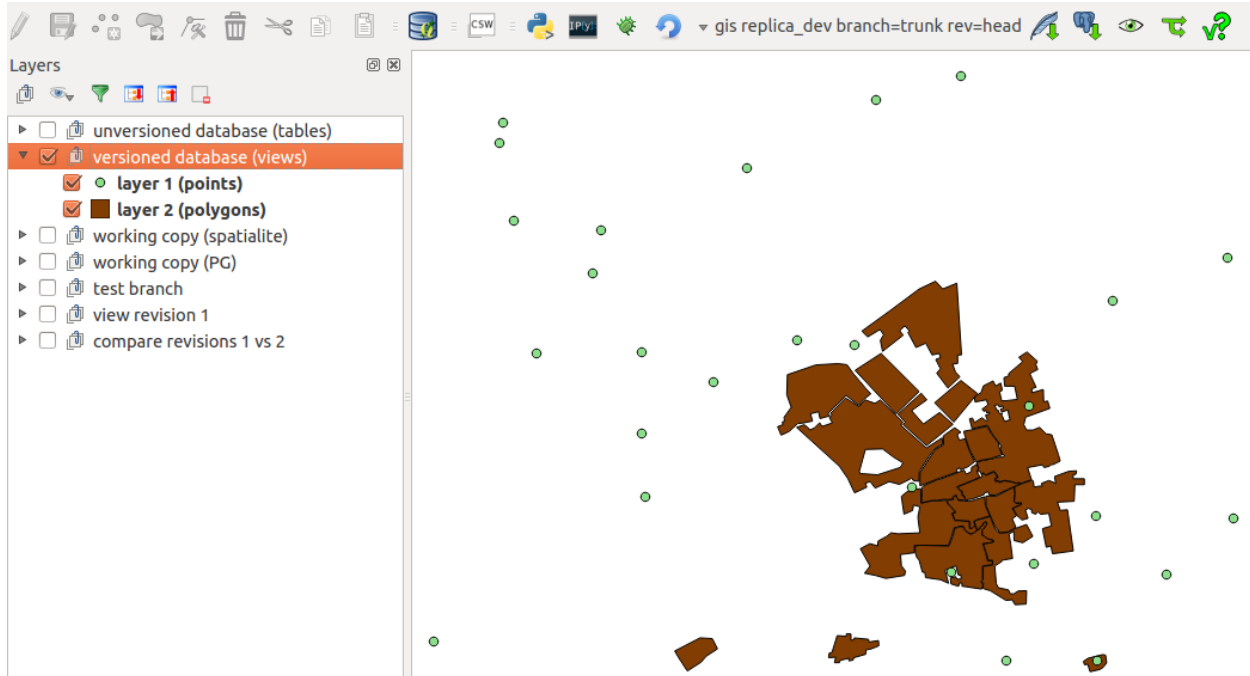
The following screenshots illustrate the various layer groups produced by the `qgis-versioning` plugin. Note that default group names (e.g. “working copy” for spatialite checkout) were modified to be more expressive.

- Unversioned database

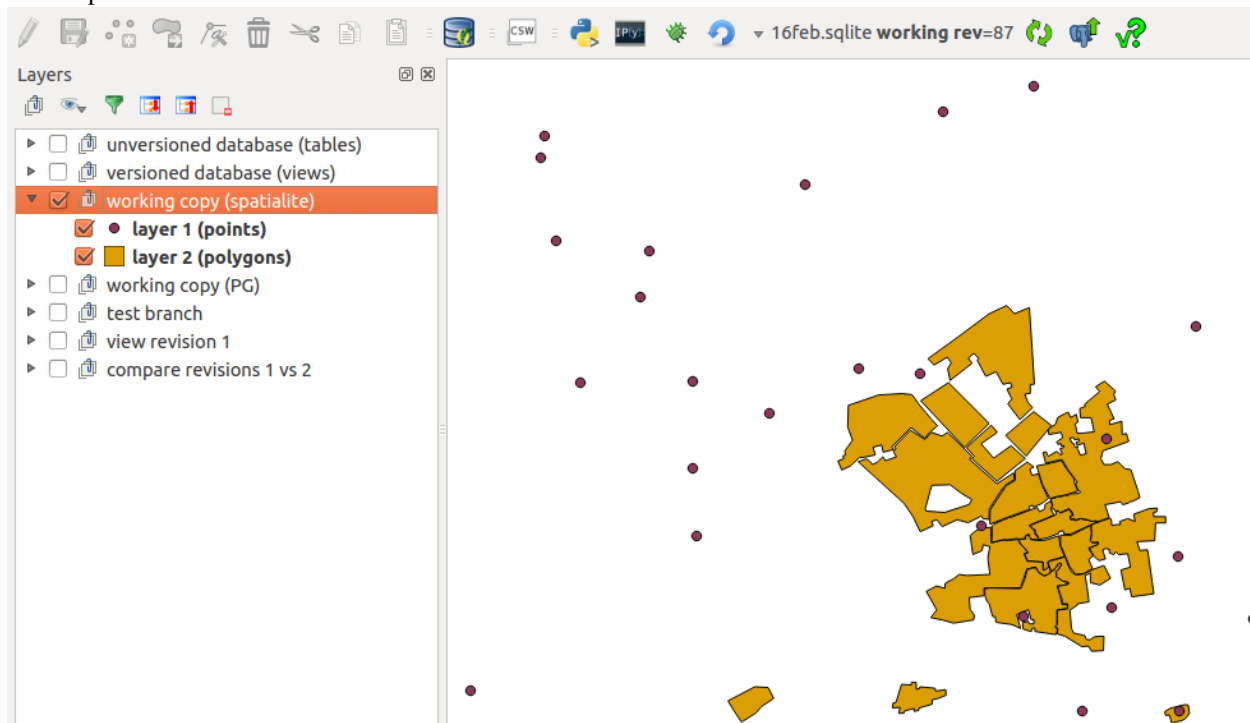


- Versioned database

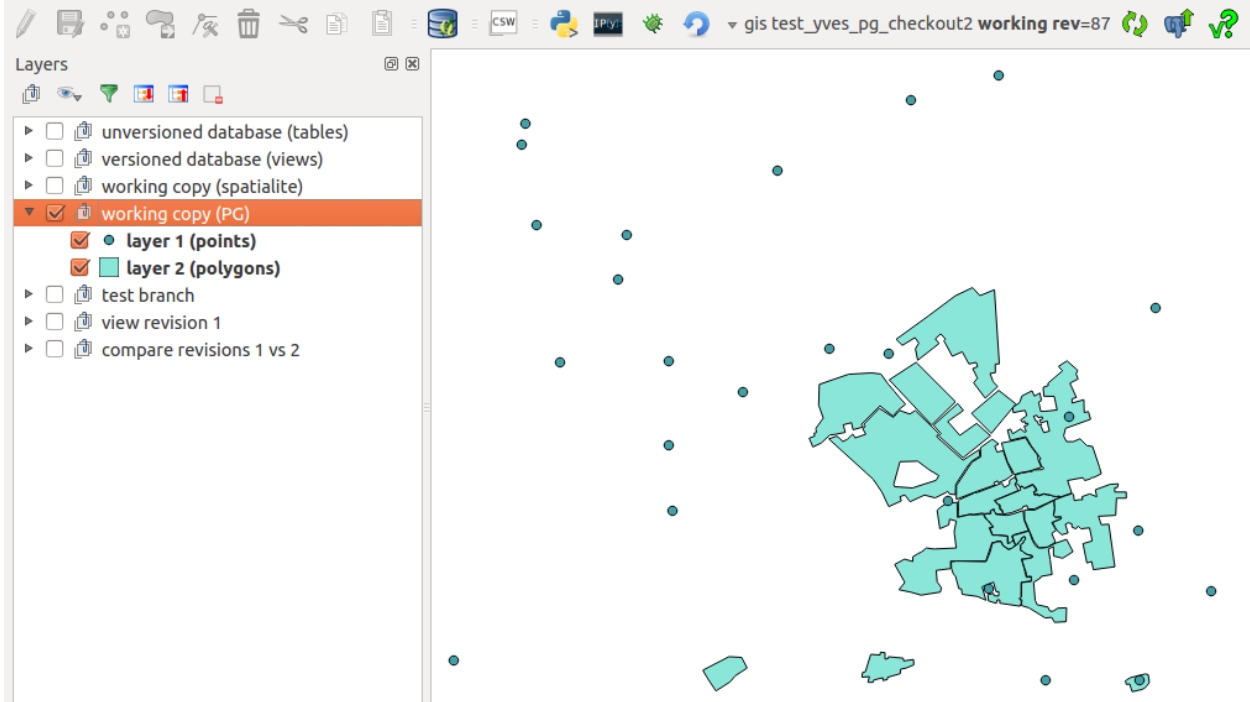




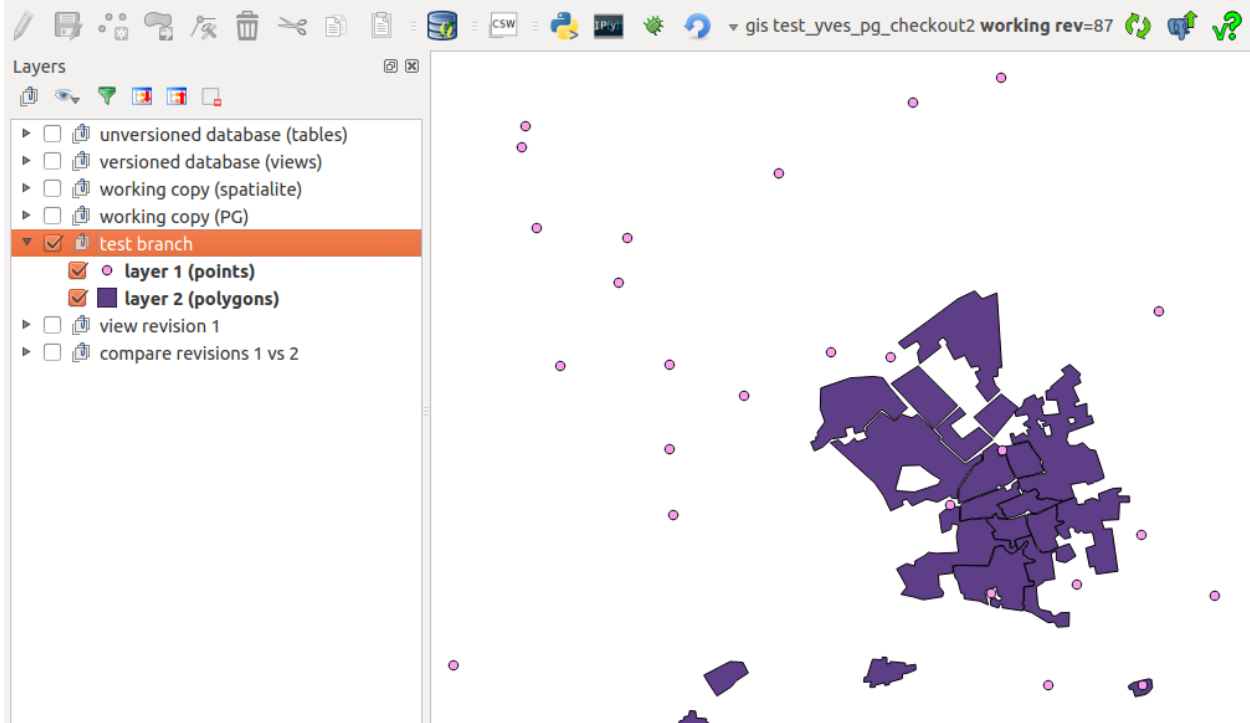
- Spatialite checkout



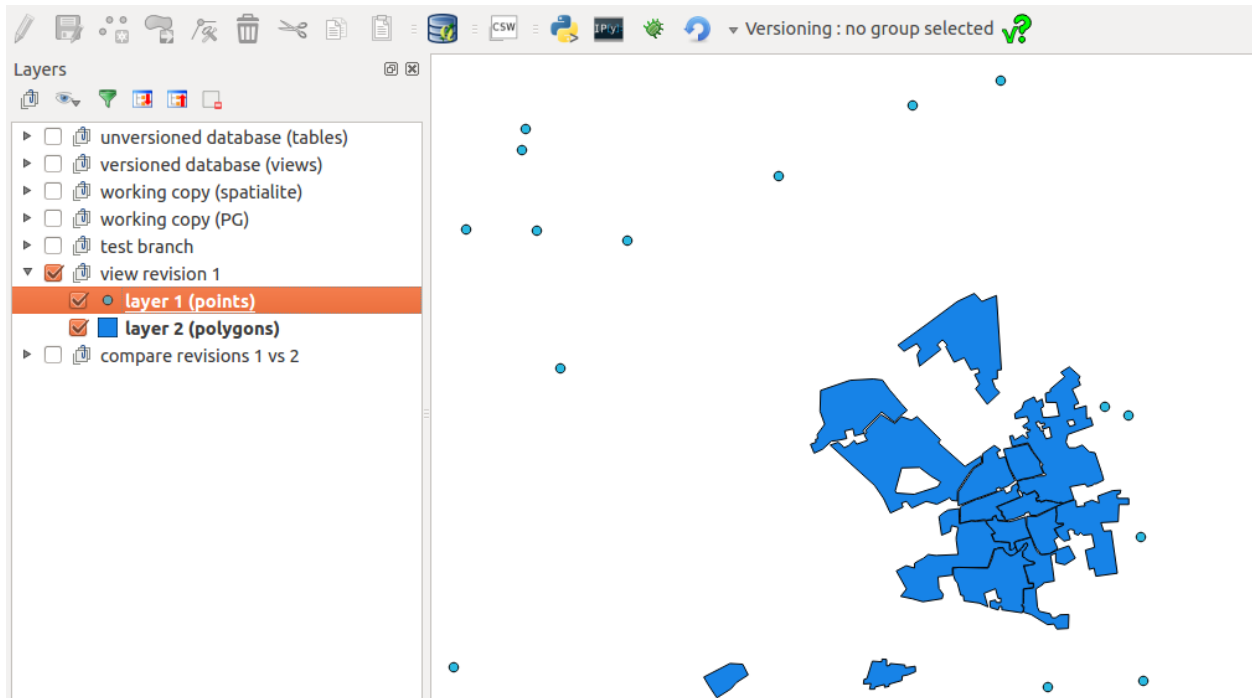
- PostGIS checkout



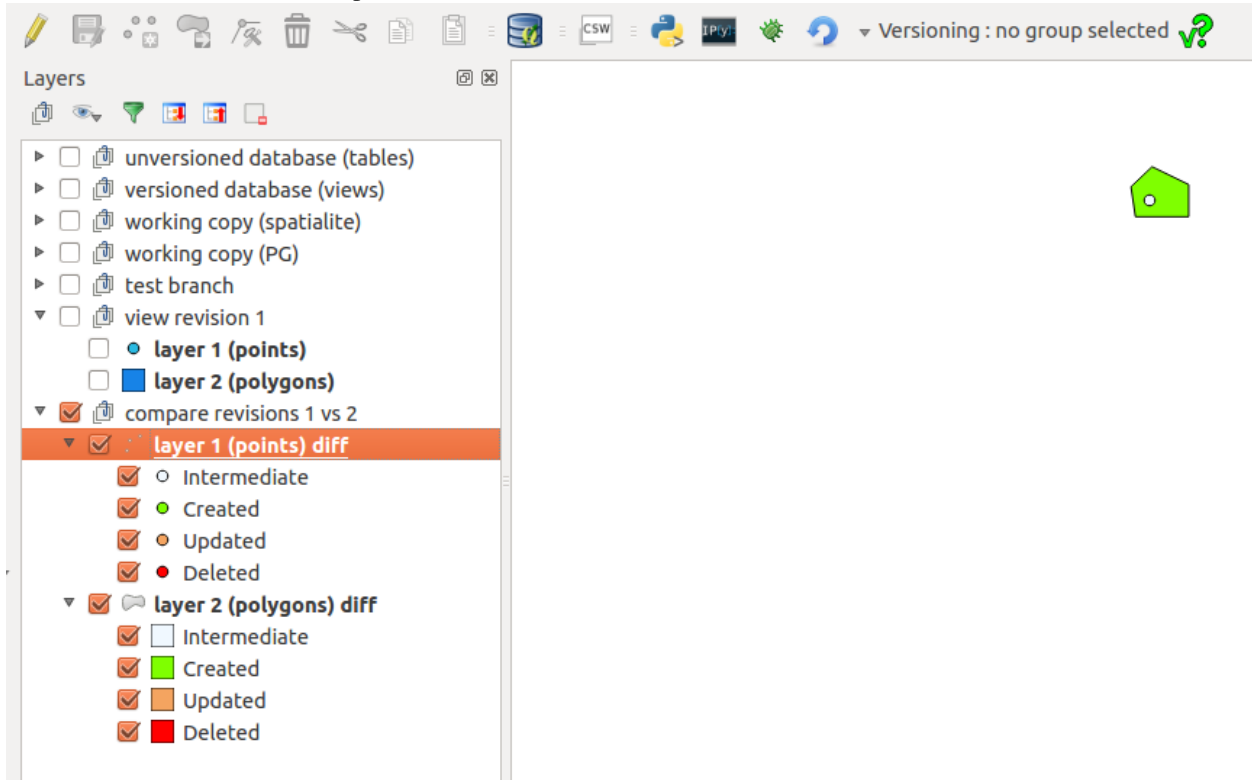
- Branch



- View revision (full mode)



- View revision (diff or comparison mode)



### 3.4 Database artifacts

Should I fill that section of just write pull-quotes specifying what happens on the DB side the in sections above ?



---

## Selecting features before checkout

---

Before checking out a local spatialite working copy, one can select features from the versioned tables to work on locally. Any layer in the group can have features selected. In the case of a layer with no features selected, the whole dataset will be checked out by the `qgis-versioning` plugin. This allows users to select only those features they are interested in editing rather than the whole dataset.

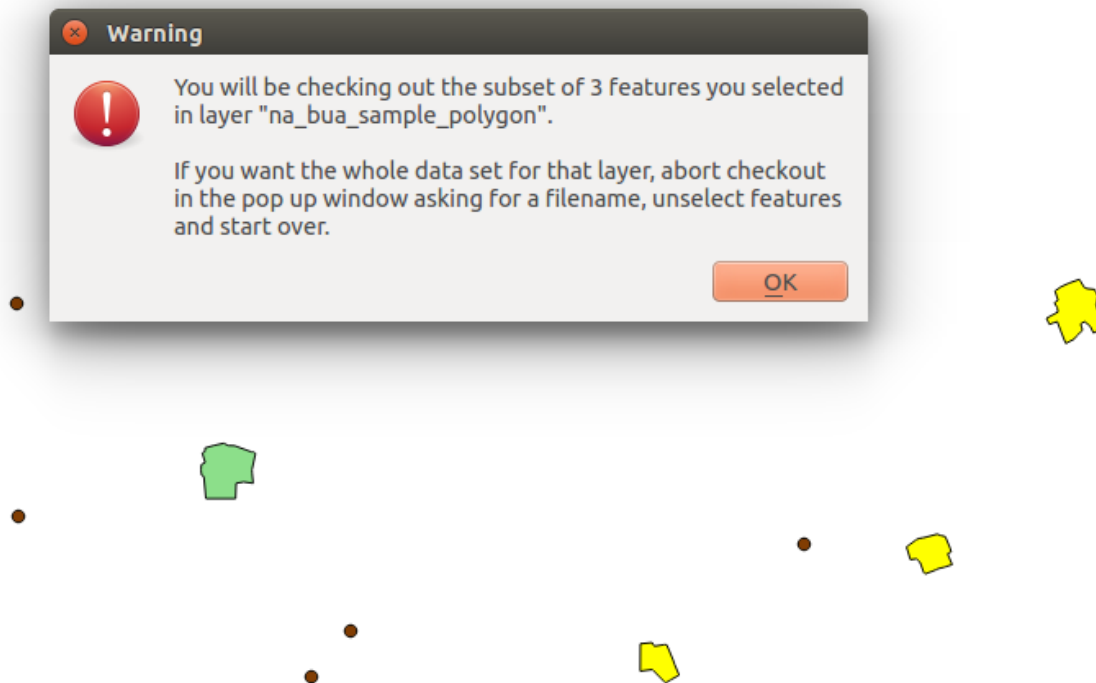
---

**Note:** Feature selection prior to checkout only applies to spatialite checkouts. It has yet to be implemented for PG checkouts.

---

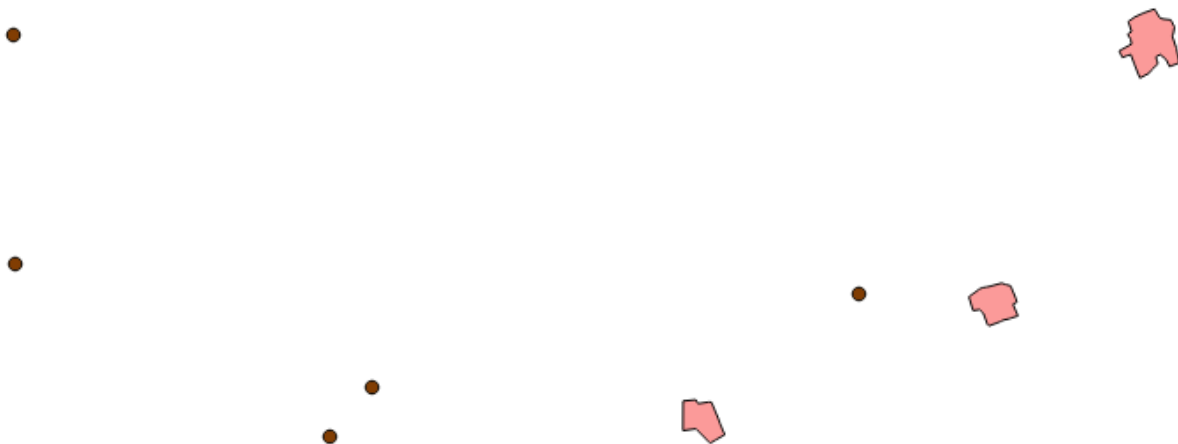
### 4.1 Procedure

- For each layer in the group, select features you want checked out. This can be done in a number of ways in QGIS.
- When ready to checkout, click on the layer group and click on the spatialite checkout button. At that point any layer with selected features will pop this warning to let the user know a subset of features will be checked out :



- Complete the rest of the default spatialite checkout workflow and check that only a subset of features was retrieved for the layers you selected features for.

In our example, only the selected polygons (yellow polygons above) and all points (since no feature selection was performed on the point layer) were checked out :



---

## Project contributors

---

While many people have contributed to the development of the `qgis-versioning` plugin, a few organizations have a key role.

### 5.1 Oslandia

Oslandia initiated this project and is the official maintainer ...

### 5.2 eHealth Africa



eHealth Africa joined the development effort in Fall 2015 ...

### 5.3 Other

Plugin contributors

Other organizations the efforts of which need to be underlined ...