# GIS Einführung mit QGIS

Christoph Rinne

11. Mai 2021

## Inhaltsverzeichnis



# 1 Datenbanken

```
select codigo, municipio, zensus_1842, zensus_1900, zensus_1950,  zensus_2011
from 'pop_balears' limit 6;
```

Tabelle 1: Einfache Abfrage für ausgewählte Felder einer Tabelle, die ersten Einträge.

| codigo | municipio | zensus_1842 | zensus_1900 | zensus_1950 | zensus_2011 |
|---|---|---|---|---|---|
| 34040707001 | Alaró | 4112 | 5982 | 3807 | 5273 |
| 34040707002 | Alaior | 4722 | 4909 | 5082 | 9450 |
| 34040707003 | Alcúdia | 1120 | 2711 | 3565 | 18914 |
| 34040707004 | Algaida | 2806 | 4091 | 3905 | 5272 |
| 34040707005 | Andratx | 4609 | 7014 | 4036 | 11234 |
| 34040707006 | Artà | 4001 | 5824 | 5496 | 7562 |

## 1.1 Vorbemerkung

In dieser Übung zu GIS mit QGIS sind (mir) zwei besonders störend Probleme aufgefallen:

- Aus Abfragen generierte Sichten oder Darstellungen von Daten müssen als neue Datei mit redundanten Daten gespeichert werden, sonst gehen diese Layer verloren und
- die Datentabellen können nur als Ganzes kopiert, nicht aber mit ausgewählten Attributen kopiert und somit schnell anderweitig verwendet werden.

Dieses Kapitel beginnt mit Übungen zu GeoPackages, SpatiaLite und der DB-Verwaltung in QGIS, am Ende steht dann die Arbeit mit der SpatiaLite-GUI.

### 1.1.1 Datenbanksysteme im Data Source Manager

Insgesamt stellt sich mir die Arbeit mit den Daten in QGIS im Vergleich zu Datenbankmanagementsystemen (DBMS) als unbefriedigend dar. Natürlich gibt es nicht nur ein DBMS und QGIS bietet bei der Datenquellenverwaltung einige Schnittstellen an. Zur Ergänzung sei erwähnt, dass ESRI (ArcGIS) auf Geodatenbanken (*geodatabases*) für den Desktop setzt, die mit MS Access kompatibel sind.

Es gibt zwei grundlegende Unterschiede bei DBMS: 1. Die für den Desktop und die Nutzung durch eine Person oder einen begrenzten Nutzerkreis in einem eher lokalen Netzwerk gedachten Systeme und davon deutlich unterschieden 2. die für Server als Dienst im Netzwerk für viele Nutzer konzipierten Systeme. Dies ist keine scharfe Grenze, denn Sie können einen SQL-Server auf ihrem PC für persönliche Zwecke betreiben oder eine Access-Datenbank erfolgreich in einem mittelständischen Betrieb einsetzen. Ein wesentlicher Aspekt neben dem System ist auch der Aufbau der Datenbank. Für GIS sind oft spezielle Ergänzungen notwendig, die nicht immer integriert sind, z.B. PostgreSQL mit PostGIS. SQL ist die Programmiersprache für das Strukturieren, Verwalten und Verändern von Daten. SQL ist textbasiert und in dieser Form auch von Menschen lesbar, gut archivierbar und trotz zahlreicher Varianten insgesamt gut interoperabel (vgl. FAIR).

- Geopackage: Ein freies und offenes DBMS auf der Basis von SQLite für den Desktop. Gegenüber SpatiaLite, an das es angelehnt ist, kann es zudem auch Rasterdaten verwalten. Der Nutzer sieht nur eine Datei, die alle Daten beinhaltet.
- SpatiaLite: Ist ein auf SQLite basierendes DBMS mit GIS-Integration speziell für den Desktop. SQLite ist frei nutzbar (*public domain*), die GIS-Funktionalität von SpatiaLite ist frei verwendbar und *open-source*. Zur Arbeit mit SpatiaLite bietet QGIS eine integrierte Anwendung, eine ergänzende SpatiaLite GUI ist aber empfehlenswert. Im Ergebnis sieht der Nutzer nur eine Datei für Vektordaten und nicht bis zu fünf Dateien je shp-Datei. Rasterdaten müssen aber getrennt verwaltet werden.
- PostgreSQL: Ein freies DBMS für Server und *open source*. Durch die Erweiterung PostGIS ist es im besonderen Maß für die Arbeit mit Geodaten geeignet. Es ist deshalb in größeren Forschungsprojekten und -verbünden beliebt. Die Wartung des Servers (Hardware), des Postgres-Servers (Dienst) und abgestimmt darauf der PostGIS-Version, sollten in ihrem Aufwand nicht unterschätzt werden. Eine Analyse des Bedarfs, der vorhandenen Kompetenz und Ressourcen ist angebracht.
- MSSQL: Microsoft SQL Server auch mit kostenfrei nutzbaren Versionen.
- Oracle: Ein traditionsreicher Anbieter von DBMS für Server, auch kostenfrei nutzbaren Versionen.
- DB2: Ist ein DBMS von IBM auch mit einer Community Edition (CE), die kostenfrei nutzbar ist.

### 1.1.2 Geopackage vs SpatiaLite

Nachfolgend ein knapper Vergleich der beiden genannten Systeme, die eigene Entscheidung hängt sicher auch vom Nutzungskonzept und der eigenen Erfahrung bzw. den Vorlieben ab. Ich konzentriere mich auf die unmittelbaren Aspekte der Anwendung, technische Details werden nicht bewertet. Zudem ist mit einer Weiterentwicklung der Systeme zu rechnen.

| pro Geopackage | contra Geopackage |
|---|---|
| - Etwas kompaktere Dateigröße, wobei GeoTIF oft größer ist als textbasierte Raster (ASC, GRD).<br>- Rasterdaten: MBTiles sind weit verbreitet.<br>- Eine Datei für alles. | - Eingeschränkte Funktionalität der SpatiaLite GUI, da Vektordaten nur als *view* zur Verfügung stehen. |

| pro SpatiaLite | contra SpatiaLite |
|---|---|
| - Eine Datei für Vektordaten.<br>- Gute GUI, u.a. mit *spatialview*-Generator.<br>- Integration von GIS-Funktionen.<br>- Integrierte Importfunktionen (dxf, shp, asc, excel).<br>- Anhängen externer Daten (virtuelle Daten).<br>- Bibliotheken für Rasterdaten, Netzwerk, Routing etc. | - Keine Rasterdaten möglich. |

Für mich ist die Arbeit mit Abfragen, speziell *spatial view* wichtig. Rasterdaten liegen oft als eine Datei vor, sind als GeoTIF oder ASC/GRD-Datei leicht austauschbar und werden mit dem Raster-Calculator effizient bearbeitet, wobei das Ergebnis jeweils leider in eine neue Datei geschrieben wird. Das Einbinden der

Rasterdaten in ein GeoPackage bringt hier keinen Vorteil. Damit ziehe ich SpatiaLite zur Zeit vor.

## 1.2 GeoPackages in QGIS

### 1.2.1 Erstellen der GPKG-Datei

Die GPKG-Datei wird leicht durch den Export von Layern aus QGIS erstellt. Wir exportieren nachfolgend die gefilterten shp-Dateien (Balearen, "CODNUT2" = 'ES53') und die Tabellen mit Daten in eine GPKG. Achten Sie jeweils auf die Angaben zum Namen und die Projektion. Abschließend folgt das DGM200 (mdt200) mit einem typischen Problem.

Aus dem Kontextmenü des Layer "Autonome Gebiete" (autonomias) wählen Sie "Exportieren -> Objekt speichern als" und im neuen Fenster setzen Sie folgende Parameter:

- Format: GeoPackage,
- Dateiname [...]: im Projektordner "mallorca.gpkg",
- Layername: autonomias (**bitte nicht die Vorgabe**),
- KBS: EPSG 4258 (original der shp),
- FID: fid (feature id),
- Geometry_Name: geom.

Die **Nomenklatur für die *feature id* (fid) und vor allem die Geometrie (geom) sollte in der Datenbank einheitlich, kurz und aussagekräftig sein**, das erleichtert die spätere Arbeit ungemein. Wiederholen Sie den Vorgang für die Provinzen (Layername: provincias), Gemeinden (Layername: municipios) und die selbst generierten Dateien comarcas der Landschaftszonen (Layername: comarcas) und mallorca-sites (Layername: sites). Exportieren Sie auch die Datentabelle pop-balears in die GPKG, bei Geometrie wählen Sie hier "keine Geometrie".

Exportieren Sie zuletzt die Rasterdatei des Höhenmodelles mdt200 bzw. DGM200 in das Geopackage mallorca. Sie sollten eine Fehlermeldung erhalten:

```
Konnte Raster nicht schreiben. Fehlercode: Datenquellenerstellung
Cannot create new dataset  D:\CAU\Lehre\20SoSe\GIS\Daten\mallorca\mallorca.gpkg:
Only Byte, Int16, UInt16 or Float32 supported
```

Öffnen Sie die Eigenschaften -> Informationen zu diesem Layer, dort finden Sie u.a. "Daten Typ Float32 - 64 Bit Fließkommazahl". Das scheint auf den ersten Blick korrekt, ist aber das Problem. Float32 (Gleitkommazahl) kann als 64 Bit oder als 32 Bit vorliegen. Die aktuellen Daten sind 64 bit, der Import ist aber nur für 32 bit Daten möglich, in der Konsequenz müssen wir also erst konvertieren. Öffnen Sie die Werkzeugkiste ("Verarbeitung -> Werkzeugkiste" oder <strg>+<alt>+<t>) und suchen Sie "konvertieren". Wählen Sie aus den GDAL-Tools "Umwandeln (Format konvertieren)": Eingabelayer: DGM200, Fortgeschrittene Parameter -> Ausgabedatentyp: Float32, Umgewandelt: In temporäre Datei. Weisen Sie der temporären Rasterdatei noch die richtige Projektion zu ("Projektion zuweisen", 25831). Wiederholen Sie den Export in die GPKG mit dieser Datei und achten Sie auf den Layernamen: mdt200.

### 1.2.2 Arbeit mit GPKG in QGIS

Für die Nutzung einer Datenbank muss stets erst eine Verbindung zu dieser eingerichtet werden. In QGIS gibt es zwei Wege: 1. über die **Datenquellenverwaltung** und 2. über die **DB-Verwaltung** ("Datenbank -> DB-Verwaltung"). Bei 1. können Sie mit [Neu] eine DPKG auswählen und anbinden, im *drop-down* vorhandene Verbindungen selektieren und mit [Verbinden] öffnen. Bei 2. wird über das Kontextmenü (rechter Mausklick) zum DBMS eine neue Verbindung eingerichtet. Alle vorhanden Verbindungen werden angezeigt und durch das Öffnen verbunden. Der gleichzeitige Zugriff über beide Wege ist möglich. In beiden Fällen können Daten aus der DPKG in das Projekt übernommen werden, bei 1. durch [Hinzufügen], bei 2. durch *drag 'n drop* oder mit Doppelklick. Damit enden die Gemeinsamkeiten.

**DB-Verwaltung von QGIS** ("Datenbank -> DB-Verwaltung") bietet schneller mehr Informationen zum jeweiligen Datenbestand und erlaubt zudem das Importieren neuer Daten sowie die Arbeit mit SQL-Anweisungen.

Bei der Datentabelle “pop-balears” fallen zwei Punkte auf: 1. Nach den “Allgemeinen Informationen” sind die Spalten 1842 bis 1920 als Text und nicht als Zahl gespeichert. 2. Das “-” im Namen führt zu Missverständnissen, bedingt Anführungszeichen und führt somit zu mehr Tipparbeit. Dazu folgende Übungen.

### 1.2.3 Einfache SQL-Anweisungen

Mit <F2> oder “Datenbank -> SQL-Fenster” öffnen Sie ein Fenster für SQL-Anweisungen. In der oberen Hälfte werden Anweisungen geschrieben und in der unteren Hälfte sehen Sie das Ergebnis. Das kleine Icon [SQL] oben links öffnet den SQL-Anweisungseditor, der die Konstruktion komplexerer Anweisungen erleichtern soll. Diesen Editor nutzen wird jetzt nicht. Schreiben Sie folgende SQL-Anweisungen nacheinander in die obere Hälfte des SQL-Fensters und betätigen Sie jeweils [Ausführen]:

- `select * from sites;`
- `select * from 'pop-balears';`
- `select municipio, 1950, '2011' from 'pop-balears';`

Die letzte Abfrage liefert folgendes Ergebnis:

| municipio | 1950 | ‘2011’ |
|---|---|---|
| Alaró | 1950 | 2011 |
| Alaior | 1950 | 2011 |
| Alcúdia | 1950 | 2011 |

Ohne Hochkomma um den Tabellennamen “pop-balears” führt die zweite Anweisung wegen des “-” zu einer Fehlermeldung. Wir können einzelne Felder wie “municipio” direkt abfragen, nur bei den Zensus die mit dem jeweiligen Jahr benannt sind bekommen wir als Ergebnis die Zahl selbst, egal ob mit oder ohne Hochkomma. Erst mit `select municipio, 1950, 'pop-balears'.'2011' from 'pop-balears';` bekommen wir für 2011 die eigentlichen Zahlen.

Um erfolgreich weiterzuarbeiten löschen Sie bitte “pop-balears” über das Kontextmenü in der Datenbank. Importieren Sie über “Tabelle -> Layer/Datei importieren” die Datei “pop_balears.csv”, achten Sie auch auf den Namen in der Datenbank “pop_balears” und setzen Sie einen Haken für die Zeichencodierung “UTF-8”. Führen Sie danach folgende Abfrage aus:

```
select codigo, municipio, zensus_1842, zensus_1900, zensus_1950,  zensus_2011
from 'pop_balears'
where municipio like "A%";
```

Tabelle 5: Abfrage für einzelne Attribute der Tabelle, Bedingung: Gemeindename beginnt mit A.

| codigo | municipio | zensus_1842 | zensus_1900 | zensus_1950 | zensus_2011 |
|---|---|---|---|---|---|
| 34040707001 | Alaró | 4112 | 5982 | 3807 | 5273 |
| 34040707002 | Alaior | 4722 | 4909 | 5082 | 9450 |
| 34040707003 | Alcúdia | 1120 | 2711 | 3565 | 18914 |
| 34040707004 | Algaida | 2806 | 4091 | 3905 | 5272 |
| 34040707005 | Andratx | 4609 | 7014 | 4036 | 11234 |
| 34040707006 | Artà | 4001 | 5824 | 5496 | 7562 |

Und zum Abschluss:

```
select m_comarca, st_area(st_transform(geom, 25831))/100000 as a_qkm
from comarcas;
```

Was haben Sie gelernt?

1. SQL-Anweisungen sind nicht grundsätzlich kompliziert, aber es gibt wichtige Syntaxregeln und Namenskonventionen, z.B. Ziffern am Anfang von oder "-" in Namen sind schlecht. Wenig beachtet, aber das ";" steht am Ende der Anweisung.
2. Wir können Daten filtern, die Syntax entspricht auffallend den Anweisungen bei Layern in QGIS.
3. Wir können gezielt Spalten (Attribute) auswählen.
4. Es gibt Funktionen, die unsere Daten verändern.

Dies sind allgemein gültige Regeln, nicht nur für die Datenbankverwaltung in QGIS. Wechseln wir nun aber zu SpatiaLite und zur SpatiaLite-GUI.

## 1.3 SpatiaLite und SpatiaLite-GUI

Wie eingangs erläutert, beruht SpatiaLite ebenfalls auf SQLite, es kann keine Rasterdaten speichern, hat eine deutlich längere Entwicklung hinter sich und bietet eine eigene GUI (graphical user interface). SpatiaLite-Datenbanken können wie DPKG durch den Export von Layern in QGIS erzeugt, gefüllt und nach der Verbindung mit QGIS-Bordmitteln genutzt werden. Bis zu diesem Punkt ergibt sich kein Mehrwert für die Nutzung. SpatiaLite erlaubt es, ***spatial views*** zu erstellen, das sind gespeicherte Abfragen mit Geometrie, die wie andere Objekte in QGIS genutzt werden können. **Das redundante Erstellen neuer Daten und der Verlust von temporär erstellten Daten wird damit vermieden**.

### 1.3.1 SpatiaLite installieren

Laden Sie sich bitte von der Website die Dateien für SpatiaLite herunter. Unten links sehen Sie die "MS Windows binaries".

1. Wählen Sie hier bei "**previous stable version**", je nach System 32bit oder 64bit. Nur bei dieser Version ist die GUI dabei.
2. Sie sehen eine drei gepackte Dateien. Laden Sie alle herunter, auch wenn wir aktuell nicht alle brauchen.
3. Zum Entpacken dieser Dateien empfehle ich die freie Software 7-zip.
4. Legen Sie im Ordner "Program Files" (64bit) oder "Program Files (x86)" einen neuen Ordner "spatialite" an und kopieren Sie alles dort hinein (Adminrechte). Da die Datei proj.db in jeder der gepackten Dateien vorliegt, belasse ich ... modules... und ...tools... in ihrem jeweiligen Unterordner. Das Programm wird nicht installiert, d.h. im Betriebssystem eingebunden, der Ordner kann also überall liegen.
5. Wenn Sie es zukünftig öfter nutzen wollen und mehr Komfort wünschen, können Sie die Datei "spatialite_gui.exe" mit dem Eintrag "An Start anheften" aus dem Kontextmenü in das Startmenü der Programme eintragen.

Zu SpatiaLite gibt es zahlreiche ergänzende Infos:

- SpatiaLite: http://www.gaia-gis.it/gaia-sins/
- SpatiaLite-GUI: https://www.gaia-gis.it/fossil/spatialite_gui/index
- SpatiaLite Wiki: https://www.gaia-gis.it/gaia-sins/spatialite_topics.html
- SpatiaLite Cookbook: http://www.gaia-gis.it/gaia-sins/spatialite-cookbook-5/index.html

### 1.3.2 SpatiaLite GUI

Starten Sie die SpatiaLite-GUI und erstellen Sie mit "Menue -> Creating a New ..." eine neue SQLite Datenbank im Projektordner, Name: mallorca.sqlite. Diese Datenbank wird unmittelbar geöffnet und verändert die GUI (s. u.). In der Iconleiste sind einige Symbole ausgegraut, wenn Sie nicht alle Teilanwendungen (Dateien) heruntergeladen haben.

Importieren Sie nacheinander die shp-Dateien der Verwaltungsgrenzen (autonomias, municipios) und die des Projektes (sites, comarcas): "Files -> Advanced -> Load Shapfile" (nutzen Sie nachfolgend das entsprechende Icon in der Iconleiste). Nach der Auswahl der shp-Datei erscheint ein Fenster, in dem Sie jeweils angepasst folgende Parameter setzen:

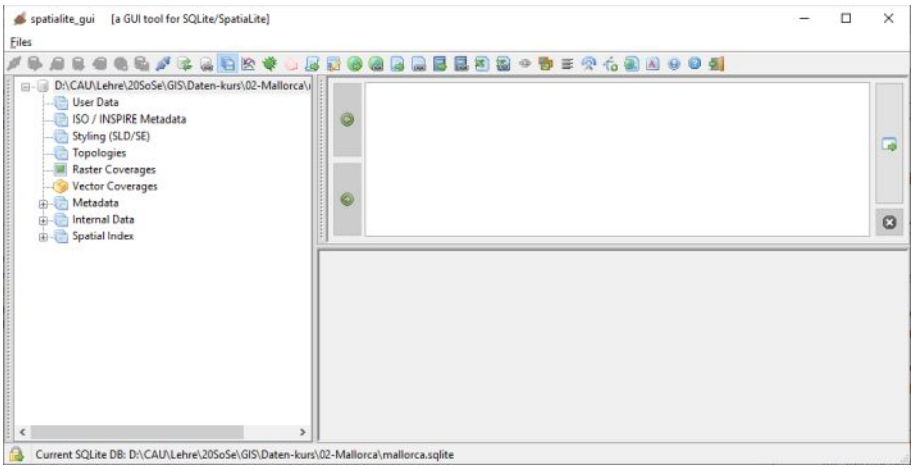


Abbildung 1: SpatiaLite-GUI

- Autonimias: Table name: autonomias, GeomColumn name: geom, SRID: Vorgabe sollte stimmen, Charset Encoding: UTF-8 (wird leider nicht automatisch ausgelesen, finden Sie in *.cpg), der Rest bleibt nach Vorgabe.
- Municipios (Gemeinden): Table name: municipios, GeomColumn name: geom, SRID: 4258, Charset Encoding: UTF-8
- Comarcas (Landschaftszonen): Table name: comarcas, GeomColumn name: geom, SRID: 25831, Charset Encoding: UTF-8.

Der Import von Text-Tabellen ist leider nicht so komfortabel wie in QGIS, z.B. bei zu überspringende Zeilen am Dateianfang. Nutzen Sie deshalb die neue Datei "pop_balears.csv" für den Import: Table Name: pop_balears, First line ....: Haken setzen, Text separator quotes: None, Column separator: Comma, Charset encoding: UTF-8 und [OK]. Falls Sie möchten, können Sie die Abfragen von oben wiederholen, achten Sie aber auf die veränderten Namen: `select municipio, zensus_1842, zensus_1930 from pop_balears`.

### 1.3.3 *view & spatial view*

Kommen wir zum zweiten wichtigen Aspekt für die Verwendung von SpataLite: gespeicherten Abfragen (*view*) mit Geometrie. Ich greife auf die frühere Abfrage zurück, wobei ich die Geometrie zur Darstellung weglasse: Wieviele Talaiots befinden sich in jeder Gemeinde?

Ich beginne mit der Auswahl der Gemeinden von Mallorca (codnut3='ES532'). Dazu nutze ich dieses Mal den "Qerry/View Composer" (Menue -> Advanced -> Querry/V...“). Register Main: Main Table: municipios, darunter bitte alle Felder markieren, Register Filter: Filter #1, Enable, Column to be filtered: codnut3, Comparison operator: =, Value: ES532, Reiter View: Create Spatial View, View name: municipios_532, Geometry Column: geom. Im Ergebnis sollten Sie einen neuen Eintrag in Ihrer DB haben, wenn nicht, führen Sie über das Kontextmenü der DB ein"refresh" durch. Verbinden Sie in QGIS im Data Source Manager die SQLite DB erneut und bestätigen Sie im aktuellen Projekt municipios_532 [Hinzufügen]. Die zuvor erstellte SQL-Anweisung erzeugt nicht nur die gefilterte Sicht auf den Datenbestand, sondern trägt zugleich die notwendigen Informationen in den Tabellen zur Verwaltung der Geometrien ein (unter Internal Data). Erst dadurch wird die *view* zur *spatial view* und in QGIS nutzbar.

Ich wiederhole den Vorgang für meine Fundplätze und Filter auf 'Talaiot...'. Main table: sites und darunter alle Spalten markieren, Filter # 1 enable, Column to be filtered: Tipo_yacim, Comparison operator: like, Value: Talaiot%, View type: Create Spatial View, View_name: talaiots, Geometry column: geom.

Nun müssen wir beiden Datenbestände räumlich verbinden, die Funktion dafür lautet: st_within(). Leider geht dies nicht mit dem *composer*.

```
create view talaiots_in_municipios as
select m.pk_uid, m.nameunit, count(t.Nr) as Talaiots
from municipios_532 as m
join talaiots as t on st_within(t.geom, st_transform(m.geom, 25831))
group by m.pk_uid;
```

1. Zeile: Erstellen der *view* talaiots_in_municipium als
2. Zeile: Es wird eine Auswahl an Feldern getroffen, dabei steht "m." für Felder aus *municipios* und "t." steht für Felder aus *talaiots*. Die Funktion count() zählt die Menge der eingetragenen Datensätze und erhält die Spaltenüberschrift "Talaiots".
3. Zeile: Als primäre Datenquelle wird *municipios_532* angegeben, die als "m" abgekürzt bezeichnet wird.
4. Zeile: Beschreibt die Verknüpfung der primären Datenquelle (m) zu *talaiots*, etikettiert als "t", mit der Funktion "st_within(t.geom, m.geom)". Dies verbindet alle Fundplatzeinträge (sites) mit der Gemeinde (municipios) in der diese liegen. Da unterschiedliche KBS vorliegen, muss vorab eine Transformation für durchgeführt werden: "st_transform()".
5. Zeile: Dieser Ausdruck gruppiert das Ergebnis nach der ID (pk_uid) der Gemeinden und bewirkt das Zählen der Funktion count() über 1 hinaus bis zur entsprechenden Menge an Talaiots je Gemeinde.

Die Abfrage braucht etwas Zeit (bei mir 6 Sek.), bitte warten Sie. Bis hierher hätte das auch in einem Rutsch ohne die ersten beiden *views* ausgeführt werden können. Die beiden *views* sind aber durchaus auch sinnvoll anderweitig einsetzbar. Da das Ergebnis nur eine Tabelle ist, müssen wir diese erneut mit unseren *municipios* und deren Geometrie verbinden und als *spatial view* speichern. Starten Sie den *composer*. Main table: talaiots_in_municipios und darunter alle Felder auswählen, Table #2 enable: municipios, Felder: PK_UID, geom, [Inner] Join, Join match #1: PK_UID zu PK_UID, View: Creat Spatial View, View name: talaiots_municipios, Geometry Column Table #2 geometries: geom.

Verbinden Sie die DB in QGIS erneut, fügen Sie die Daten zum Projekt hinzu und warten Sie bis der Vorgang abgeschlossen ist (ca. 12 Sek. bei mir). Das ist natürlich ärgerlich, zumal dies auch bei der Zuweisung der Symbologie (kategorisiert) erneut auftritt. Die Daten sind aber 1. nicht redundant hinterlegt, 2. wirken sich Änderungen an den Fundplätzen direkt auf das Ergebnis aus, die Darstellung ist also stets aktuell und 3. ist die Anweisung in der DB dauerhaft gespeichert und damit dokumentiert.