7. Treffen der Schweizerischen QGIS-Anwender 15. Juni 2016 / Bern

Vom INTERLIS-Datenmodell zur QGIS-Fachschale

Oliver Jeker, AGIS Service Center, Kanton Aargau
Stefan Ziegler, Amt für Geoinformation, Kanton Solothurn

Inhaltsverzeichnis

1 Allgemeines

1.1 Einleitung

QGIS und seine Formularfunktionen bieten in Kombination mit *ili2pg* und *PostgreSQL/PostGIS* eine Plattform für die einfache Erstellung von sogenannten Fachschalen. Diese Fachschalen dienen hauptsächlich der Datenerfassung in einem speziellen Datenmodell. Das konzeptionelle Datenmodell liegt häufig in der Modellierungssprache INTERLIS vor.

Der Workshop zeigt anhand eines kantonalen Nutzungsplanungs-Datenmodells wie die drei Programme sinnvoll für die Datenerfassung und den Datenexport kombiniert werden können.

Als Datenreferenz dient die amtliche Vermessung im Bezugsrahmen LV95.

1.2 Voraussetzungen

Das Funktionieren und Zusammenspielen der benötigten Programme ist unbedingt vor dem Workshop zu testen.

1.2.1 Programme

- QGIS Master (nightly build) oder QGIS 2.14 (mit ein paar Abstrichen)
- PostgreSQL/PostGIS
- pgAdmin3
- Java

- ili2pg-3.0.3

1.2.2 Kenntnisse

Es werden Kenntnisse in *QGIS*, *PostgreSQL/PostGIS* und INTERLIS vorausgesetzt. Ein Grundverständnis im Bereich Nutzungsplanung ist auch nicht verkehrt.

1.2.3 Datengrundlagen

Sämtliche für den Workshop benötigten Daten können hier heruntergeladen werden:

- https://git.sogeo.services/stefan/qgis-ili2pg-workshop
- https://git.sogeo.services/stefan/qgis-ili2pg-workshop/archive/master.zip

1.2.4 Einstellungen

QGIS

Bezugsrahmen Sämtliche Daten liegen im Bezugsrahmen LV95 vor oder werden in diesem Bezugsrahmen erfasst. Die KBS-Einstellungen in *QGIS* sind unter **Optionen > KBS** entsprechend vorzunehmen, damit jedes neue Projekt mit dem korrekten EPSG-Code (2056) gestartet wird. **'Spontanreprojektion' nicht einschalten** ist zu wählen.

Darstellung **Optionen > Darstellung > Geometrievereinfachung** für neue Layer voreinstellen ist nicht zu aktivieren.

Digitalisierung **Optionen > Digitalisierung > Objektfang**: Sinnvolle Werte für **Fangtoleranzvorgabe** und **Suchradius für Stützpunktbearbeitung** sind zu wählen, z. B. 20 Pixel.

Datenquellen Unter **Optionen > Datenquellen > Datenquellenbehandlung** muss die Option **Ausdrücke wenn möglich serverseitig ausführen** aktiviert werden. Zusätzlich muss unter **Projekt > Projekteeigenschaften > Data Sources** die Option **Evaluate default values on provider side** aktiviert sein.

PostgreSQL

Es muss ein Benutzer vorhanden sein, der Schemen und Tabellen erzeugen darf, z. B.:

```
CREATE ROLE stefan LOGIN NOSUPERUSER INHERIT CREATEDB
   NOCREATEROLE NOREPLICATION;
```

2 ili2pg: Daten importieren und umbauen

2.1 Ziele

- INTERLIS-Daten in die PostgreSQL-Datenbank importieren und umbauen
- Optionen von *ili2pg* kennenlernen
- O/R-Mapping (Abbildung INTERLIS → *PostgreSQL*)

2.2 ili2pg

Gemäss Selbstdeklaration:

- Translates INTERLIS 2 data model definitions to an PostGIS schema.
- Loads INTERLIS 2 data into an PostGIS database.
- Extracts INTERLIS 2 data from an PostGIS database.

INTERLIS 1 (ITF) wird ebenfalls unterstützt. INTERLIS/GML (eCH-0118 Version 1) kann nur exportiert werden.

Ili2pg setzt Java 1.6 voraus. Am einfachsten lässt sich *ili2pg* via Terminal / Kommandozeile bedienen. Es steht zwar ein einfaches GUI zur Verfügung, jedoch sind damit nicht alle Optionen verwendbar.

Die Dokumentation liegt in Form einer PDF-Datei der Software bei *(docs/ili2pg.pdf)*.

2.3 Programmoptionen

Um sämtliche Programmoptionen anzeigen zu lassen, reicht folgender Befehl:

```
java -jar ili2pg.jar --help
```

Es erscheint auf dem Bildschirm eine grosse Anzahl an Optionen, mit denen man das Programm auf vielfache Weise steuern kann. Sämtliche Optionen werden in der Software-Dokumentation beschrieben.

Im Verlauf des Workshops werden wir viele der Optionen kennen lernen und verwenden.

2.4 Daten der amtlichen Vermessung importieren

Als Referenz für die Erfassung der Ortsplanung dient die amtliche Vermessung. Die Daten der Gemeinde liegen im Bundesmodell (DM01AVCH24LV95D[1]) im Bezugsrahmen LV95 vor und können mit folgendem Befehl in die Datenbank importiert werden:

```
java -jar ili2pg.jar --dbhost localhost --dbport 5432 --
   dbdatabase xanadu2 --dbusr stefan --dbpwd ziegler12 --
   nameByTopic --sqlEnableNull --createFkIdx --createGeomIdx --
   defaultSrsAuth EPSG --defaultSrsCode 2056 --modeldir http://
   models.geo.admin.ch --models DM01AVCH24LV95D --dbschema
   av_2495 --import daten/av/ch_249500_lv95.itf
```

`dbhost`
`dbport`
`dbdatabase`
`dbusr`
`dbpwd`
	Datenbankverbindungsparameter. Host und Port haben Standardwerte («localhost» und «5432») und müssen nicht explizit angegeben werden.

[1] http://models.geo.admin.ch/V_D/DM.01-AV-CH_LV95_24d_ili1.ili

`nameByTopic`	Für die Tabellennamen werden die qualifizierten INTERLIS-Klassennamen (Topic.Class) verwendet und in gültigte Datenbanknamen (Topic_Class) umgewandelt.
`sqlEnableNull`	Erstellt keine «NOT NULL»-Anweisungen bei obligatorischen INTERLIS-Attributen.
`createFkIdx`	Erstellt für jede Fremdschlüsselspalte einen Index.
`createGeomIdx`	Erstellt für Geometriespalten in der Datenbank einen räumlichen Index.
`defaultSrsAuth` `defaultSrsCode`	SRS-Authority und -Code. Standardmässig sind «EPSG» und «21781» (LV03). Achtung: Es findet keine Projektion oder Transformation statt. Einzig die Tabellen und die einzelnen Geometrien erhalten die korrekte Metainformation.
`modeldir`	Verzeichnis mit den INTERLIS-Modellen. Verschiedene Verzeichnisse können mit Semikolon getrennt werden (innerhalb von Anführungszeichen).
`models`	Das Datenmodell des zu importierenden Datensatzes.
`dbschema`	Der Namen des zu erzeugenden Schemas in dem alle Tabellen erzeugt werden.
`import`	Daten werden importiert.
`.../ch_249500_lv95.itf`	Pfad zum Datensatz, der importiert werden soll.

Tabelle 2.1: Verwendete ili2pg-Optionen

Nicht verwendete Optionen:

`createscript`	Datenbankverbindungsparameter. Host und Port haben Standardwerte («localhost» und «5432») und müssen nicht explizit angegeben werden.
`createEnumColAsItfCode`	Bildet bei Aufzählungsattributen den Aufzählwert als ITF-Code ab, z.b. statt «Gebaude» wird der «0» (Integer) in die Datenbank geschrieben.

`maxNameLength <length>`	Die maximale Länge der Namen für Datenbankelemente (Tabellennamen, Spaltennamen, etc.). Standard ist 60. Sind die INTERLIS-Namen länger, werden sie gekürzt.
`strokeArcs`	Segmentiert die Kreisbgogen beim Datenimport.
`createGeomIdx`	Erstellt für Geometriespalten in der Datenbank einen räumlichen Index.
`skipPolygonBuilding`	Bei ITF-Dateien werden keine Polygone gebildet. Praktisch für Fehlersuche.
`importTid`	Liest die Transferidentifikation aus der Transferdatei in eine zusätzliche Spalte «T_Ili_Tid». Der Primärschlüssel jeder Tabelle («T_Id») entspricht nicht der Transferidentifikation der Transferdatei, sondern wird von der Datenbank resp. ili2pg selber verwaltet.
`t_id_Name <name>`	Definiert den Namen für die interne technische Schlüsselspalte (Primärschlüssel) in jeder Tabelle. Standard ist «T_Id».
`createFk`	Erzeugt eine Fremdschlüsselbedingung bei Spalten, die Records in anderen Tabellen referenzieren.
`createUnique`	Erstellt für INTERLIS-UNIQUE-Constraints in der Datenbank UNIQUE-Bedingungen (sofern abbildbar).
`oneGeomPerTable`	Erzeugt Hilfstabellen, falls in einer Klasse mehr als eine Geometrie-Attribut vorhanden ist. Jede Datenbanktabelle erhält somit nur eine Geometriespalte..
`log <logfile>`	Schreibt die Log-Meldungen in eine Datei.
`trace`	Erzeugt zusätzliche Log-Meldungen.

Tabelle 2.2: Zusätzliche ili2pg-Optionen

Einige weitere interessante und wichtige Optionen werden in den nachfolgenden Bei-

spielen verwendet und erläutert. Der Konsolen-Output von *ili2pg* liefert während des Importes wichtige Informationen. Ein falsches Passwort wird mit folgender Meldung quittiert:

```
failed to get db connection
FATAL: password authentication failed for user "stefan"
```

Konnte der Import der Daten erfolgreich durchgeführt werden, erscheint eine Statistik der importierten Elemente und als letzte Zeile «Info: ...import done». In der Datenbank ist nun das Schema «av_2495» mit 145 Tabellen vorhanden sein:

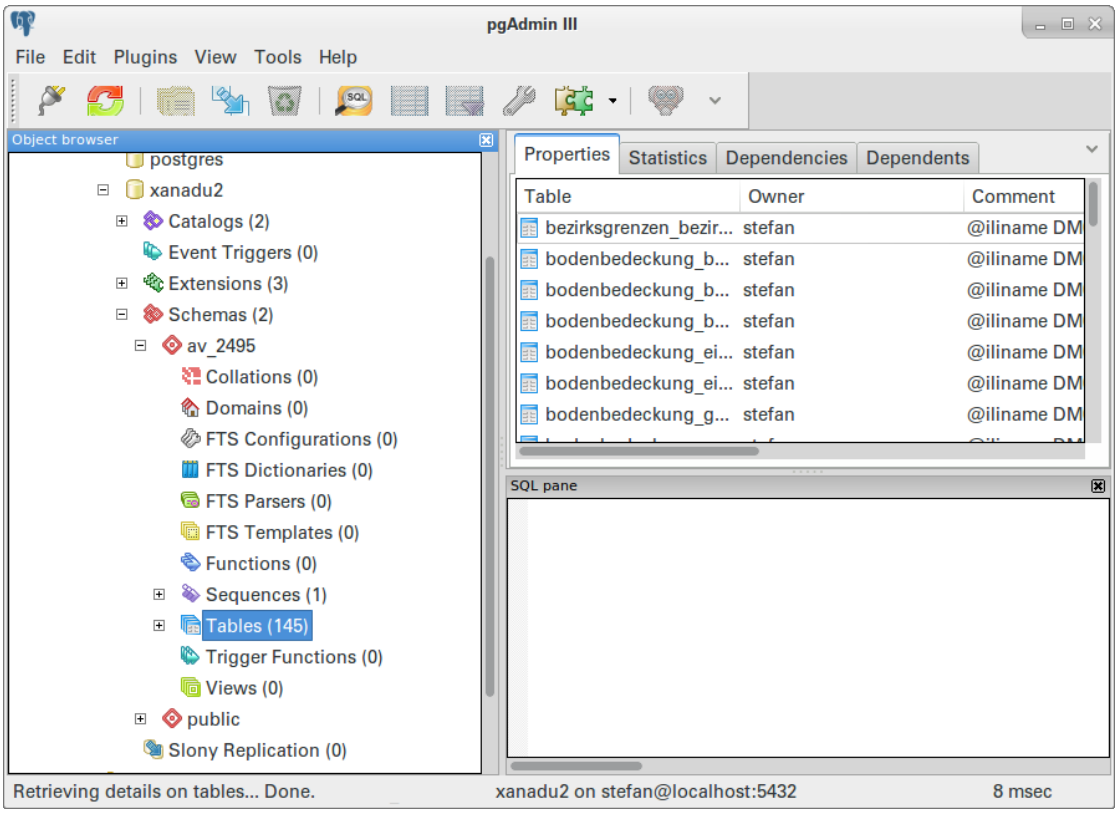
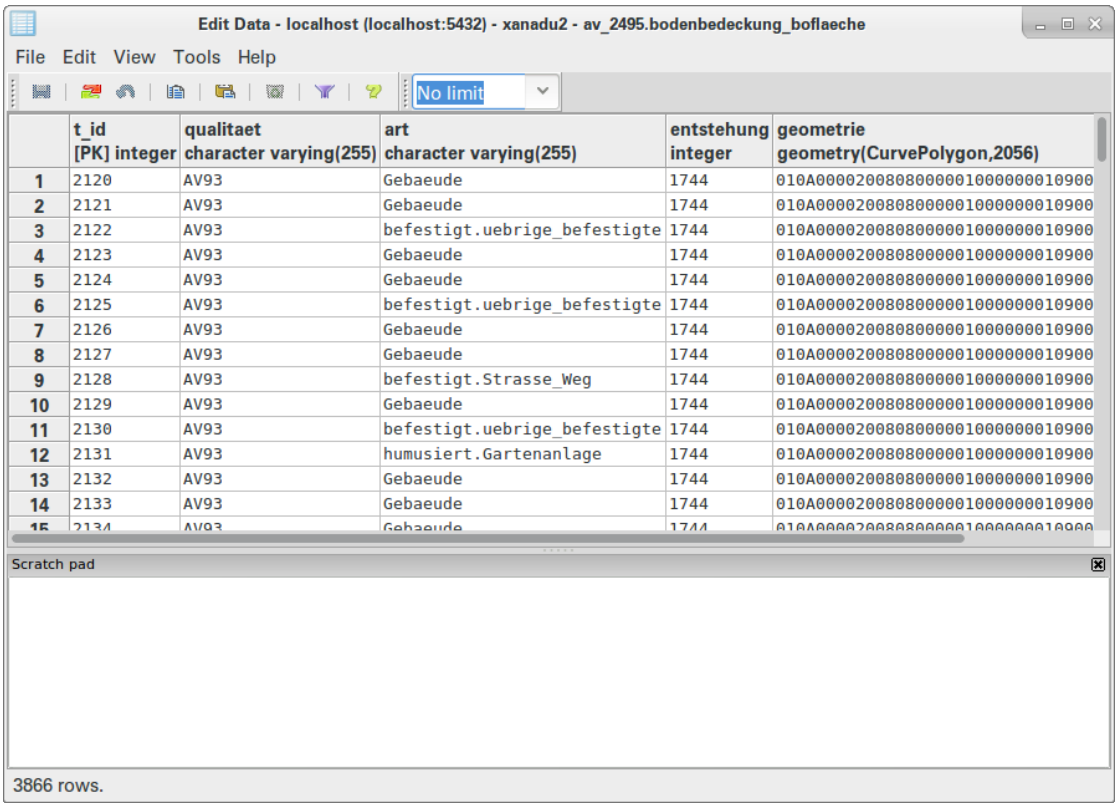


Abbildung 2.1: AV-Daten in das Schema «av_2495» importiert.

Die Daten werden wir erwartet importiert. Die Tabelle «BoFlache» des Topics «Bodenbedeckung» heisst in der Datenbank «bodenbedeckung_boflache». Die Aufzählytpen

werden qualifiziert gespeichert («a.b.c»). Der Geometrietyp ist «CurvePolygon» und der Bezugsrahmen ist LV95:



	t_id [PK] integer	qualitaet character varying(255)	art character varying(255)	entstehung integer	geometrie geometry(CurvePolygon,2056)
1	2120	AV93	Gebaeude	1744	010A0000200808000001000000010900
2	2121	AV93	Gebaeude	1744	010A0000200808000001000000010900
3	2122	AV93	befestigt.uebrige_befestigte	1744	010A0000200808000001000000010900
4	2123	AV93	Gebaeude	1744	010A0000200808000001000000010900
5	2124	AV93	Gebaeude	1744	010A0000200808000001000000010900
6	2125	AV93	befestigt.uebrige_befestigte	1744	010A0000200808000001000000010900
7	2126	AV93	Gebaeude	1744	010A0000200808000001000000010900
8	2127	AV93	Gebaeude	1744	010A0000200808000001000000010900
9	2128	AV93	befestigt.Strasse_Weg	1744	010A0000200808000001000000010900
10	2129	AV93	Gebaeude	1744	010A0000200808000001000000010900
11	2130	AV93	befestigt.uebrige_befestigte	1744	010A0000200808000001000000010900
12	2131	AV93	humusiert.Gartenanlage	1744	010A0000200808000001000000010900
13	2132	AV93	Gebaeude	1744	010A0000200808000001000000010900
14	2133	AV93	Gebaeude	1744	010A0000200808000001000000010900
15	2134	AV93	Gebaeude	1744	010A0000200808000001000000010900

Abbildung 2.2: Tabelle «BoFlaeche» des Topics «Bodenbedeckung».

Hätte man die Option `--createEnumColAsItfCode` verwendet, würden statt der qualifizierten Werte bei den Attributen «qualitaet» und «art» nur die entsprechenden Integer-Werte stehen. Mit der Option `--strokeArcs` wäre der Geometrietyp nicht «CurvePolygon», sondern «Polygon».

Neben den Tabellen des INTERLIS-Datensatzes, werden auch verschiedene Meta-Tabellen angelegt. Diese beginnen alle mit dem Namen «t_ili2db_*». Die Tabelle «t_ili2db_model» beinhaltet sämtliche für den Import notwendigen INTERLIS-Datenmodelle. Bei INTERLIS 2 sind in der Tabelle verschiedene Records vorhanden.

Die Tabelle «t_ili2db_classname» verwaltet das Mapping zwischen dem qualifizierten INTERLIS-Klassennamen («iliname») und dem Tabellennamen in der Datenbank («sqlname»).

Eine ähnliche Funktion für Attributnamen übernimmt die Tabellen «t_ili2db_attrname». Gut sichtbar sind die Änderungen an Attributnamen, die gleich reservierter Namen der Datenbank sind. So wird das Attribut «Name» zu «aName» gemappt.

2.5 Daten umbauen

Die Daten können nun in *QGIS* angezeigt werden. Um aber sie auch sinnvoll darstellen zu können, müssen wir die Daten noch umbauen. Die Daten liegen normalisiert in der Datenbank vor, so sind z. B. die benötigten Informationen um Grundstücke sauber zu beschriften in zwei Tabellen gespeichert. Es gibt verschiedene Möglichkeiten dies direkt in *QGIS* zu machen:

Die erste Möglichkeit ist das Verknüpfen (Joinen) der Layer in QGIS: **Layer > Eigenschaften > Verknüpfungen**

Die elegantere Variante sind QGIS-Expressions: Die Expressions kann man für diesen Einsatzzweck in einem virtuellen Feld oder direkt in den Beschriftungfunktionen verwenden. Mit einer Kombination von Expressions kann man auf Attributwerte anderer Layer zugreifen. In unserem Fall – wo wir die Grundstücke beschriften wollen – sind wir darauf angewiesen die Grundstücksnummer aus dem Layer «liegenschaften_grundstueck» zu lesen. Die Platzierungsattribute («ori», «hali», «vali») sind im zu beschriftenden Layer («liegenschaften_grundstueckpos.pos») vorhanden. Die Expression sieht wie folgt aus:

```
attribute(get_feature('liegenschaften_grundstueck','t_id',
grundstueckpos_von),'nummer')
```

Das Ganze hat aber einen gewaltigen Nachteil: Performance. Bei einigen wenigen Objekten mag das noch funktionieren. Bei vielen Objekten, die beschriftet werden wollen, macht es keinen Spass mehr.

Die old-school Methode ist das Erstellen von Views in der Datenbank. Ist es ein Datenumbau, der nicht nur temporär ist, sondern mit grosser Wahrscheinlich immer wieder und von vielen anderen Benutzern verwendet wird, sind Views für solche Anwendungsfälle sehr praktikabel.

Da wir neben den Grundstücksnummern auch noch die Strassennamen und die Hausnummer beschriften wollen, sind insgesamt drei Views *(db/av/views.sql)* notwendig:

```
CREATE OR REPLACE VIEW
     av_2495.v_liegenschaften_grundstueckpos AS
SELECT p.*, g.nummer
FROM av_2495.liegenschaften_grundstueck as g,
     av_2495.liegenschaften_grundstueckpos as p
WHERE g.t_id = p.grundstueckpos_von;

ALTER TABLE av_2495.v_liegenschaften_grundstueckpos
OWNER TO stefan;
```

```
CREATE OR REPLACE VIEW
     av_2495.v_gebaeudeadressen_lokalisationsnamepos AS
SELECT p.*, n.atext
FROM av_2495.gebaeudeadressen_lokalisationsname as n,
     av_2495.gebaeudeadressen_lokalisationsnamepos as p
WHERE n.t_id = p.lokalisationsnamepos_von;

ALTER TABLE av_2495.v_gebaeudeadressen_lokalisationsnamepos
OWNER TO stefan;
```

```
CREATE OR REPLACE VIEW
     av_2495.v_gebaeudeadressen_hausnummerpos AS
SELECT p.*, g.hausnummer
FROM av_2495.gebaeudeadressen_gebaeudeeingang as g,
     av_2495.gebaeudeadressen_hausnummerpos as p
WHERE g.t_id = p.hausnummerpos_von;

ALTER TABLE av_2495.v_gebaeudeadressen_hausnummerpos
OWNER TO stefan;
```

Folgende Datenbanktabellen und -views können nun in *QGIS* geladen werden und mit den entsprechenden Stilen[2] symbolisiert werden:

- bodenbedeckung_boflaeche / av_boflaeche.qml
- liegenschaften_liegenschaft / av_liegenschaft.qml
- v_liegenschaften_grundstueckpos.pos / av_grundstueckpos.qml
- v_gebaeudeadressen_hausnummerpos / av_hausnummerpos.qml
- v_gebaeudeadressen_lokalisationsnamepos.pos / av_lokalisationsnummerpos.qml

Die fünf Layer sind zwecks besserer Übersicht zu gruppieren («Amtliche Vermessung»). Das Endresultat sollte so aussehen:

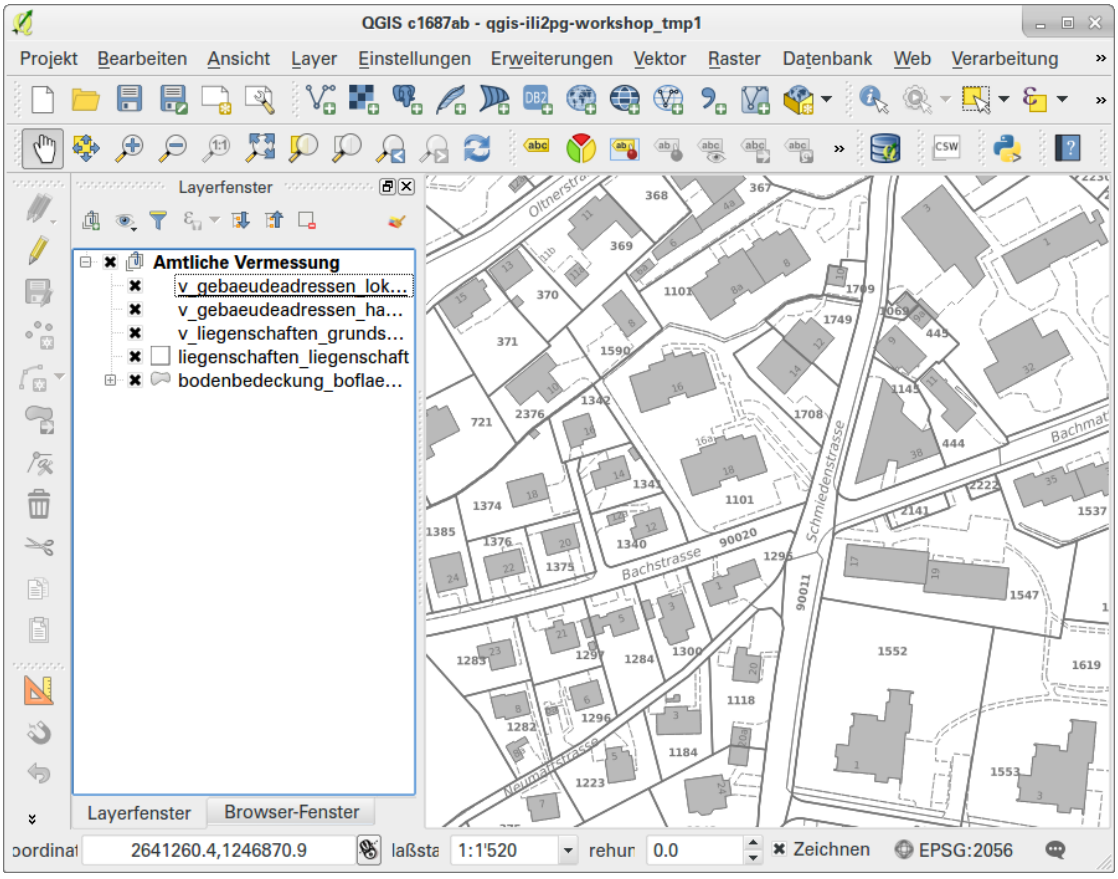


Abbildung 2.3: Darstellung der amtlichen Vermessung in QGIS.

[2] qgis/qml/av_*.qml

Zur Kontrolle: Der EPSG-Code muss 2056 sein und die Koordinaten müssen eine führende 2 resp. 1 haben.

Für die korrekte Darstellung des Planes für das Grundbuch fehlen natürlich noch viele Layer. Um die Daten als Grundlage zur Erfassung der Nutzungsplanung zu verwenden, reichen die dargestellten Informationen aus.

2.6 Ausblick: Daten exportieren

Richtig spannend wird der Datenexport erst, wenn wir die neu erfassten Daten im kantonalen Nutzungsplanungs-Datenmodell exportieren wollen. Als Vorgriff werden wir trotzdem die amtliche Vermessung wieder in eine INTERLIS-Transferdatei exportieren. Der Befehl ist noch einfacher als der Import selber:

```
java -jar ili2pg.jar --dbhost localhost --dbport 5432 --
   dbdatabase xanadu2 --dbusr stefan --dbpwd ziegler12 --models
    DM01AVCH24LV95D --dbschema av_2495 --export daten/av/
   ch_249500_lv95_export.itf
```

Die meisten Informationen, wie z. B. auch das Datenmodell selber, sind ja in den Metatabellen («t_ili2db_*) gespeichert. Damit sind viele Optionen, die wir beim Import verwendet haben, nicht mehr notwendig. Anstelle von `--import` müssen wir `--export` verwenden und einen neuen Dateinamen wählen.

Falls der Export erfolgreich funktioniert hat, erscheint auf der Konsole «Info ...export done». Ebenfalls wie beim Import, wird beim Export eine Statistik der exportierten Elemente aufgelistet.

Die exportierte INTERLIS-Transferdatei ist nicht 100%-modellkonform. Die Linienattribute gehen verloren. Dieses Sprachkonstrukt wird es bei INTERLIS 2.4 mit grosser Wahrscheinlichkeit aber nicht mehr geben.

3 O/R-Mapping

3.1 Ziele

- «Furcht» vor INTERLIS 2 und objektorientiertem Modellieren verlieren
- O/R-Mapping von *ili2pg* kennenlernen

3.2 Objektrelationale Abbildung

In INTERLIS 1 wird relational modelliert. Diese Modelle können – wie wir gesehen haben – einfach und 1:1 in der Datenbank abgebildet werden. In INTERLIS 2 kann man objektorientiert modellieren. Die Abbildung in der Datenbank wird dadurch zwar nicht einfacher aber auch nicht unmöglich.

Wichtig: Bei dieser sogenannten objektrelationalen Abbildung handelt es sich nicht um eine Erfindung von INTERLIS oder der Geo-Welt. Das Prinzip ist als Technik der Softwareentwicklung «uralt» und bewährt.

Es gibt verschiedene Möglichkeiten die Objekte aus INTERLIS 2 in die relationale Datenbank abzubilden. *Ili2pg* unterstützt selbst verschiedene Varianten. Neben der Abbildung von Vererbungshierachien müssen auch Struktur-Attribute und Assoziationen in der Datenbank abgebildet werden.

3.3 Schema in Datenbank vorbereiten

Um die Nutzungsplanung in einer Gemeinde zu erfassen, müssen wir zuerst die leeren Tabellen in der Datenbank anlegen. Ili2pg kennt dafür die Option –schemaimport. Damit lassen sich anhand eine INTERLIS-Modelles das Schema und die leeren Tabellen in der Datenbank erzeugen. Also genau das was wir brauchen, um anschliessend die Nutzungsplanung der Gemeinde zu erfassen.

Das Datenmodell «SO_Nutzungsplanung_20160427»[1], das wir verwenden, befindet sich zurzeit in Bearbeitung und ist noch nicht definitiv. Es ist aber passend für die Demonstration der Abbildungsregeln in INTERLIS 2 geschrieben und verwendet Assoziationen (siehe Abbildung 3.1) und Vererbungen (siehe Abbildung 3.2). Was fehlt sind unter anderem Struktur-Attribute und Multi-Geometrien.

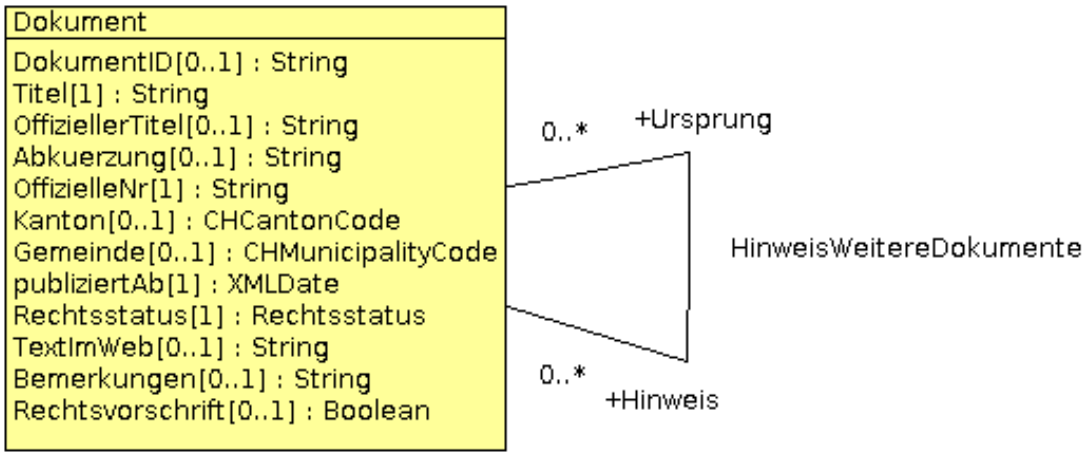


Abbildung 3.1: Assoziation

[1] datenmodell/ili2/SO_Nutzungsplanung_2016-04-27.ili

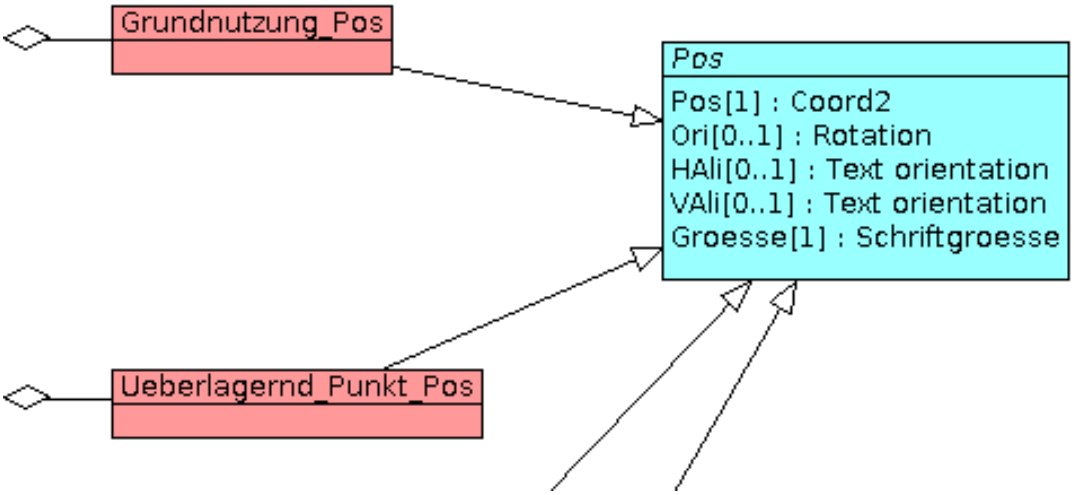


Abbildung 3.2: Vererbung

Der ili2pg-Befehl ist praktisch identisch dem Import der Daten der amtlichen Vermessung. Einzig, dass wir jetzt keine Daten importieren müssen, sondern bloss die leeren Tabellen anlegen wollen und dass wir zwei Modell-Repositories (lokales und Bundes-Repository) angeben müssen:

```
java -jar ili2pg.jar --dbhost localhost --dbport 5432 --
   dbdatabase xanadu2 --dbusr stefan --dbpwd ziegler12 --
   nameByTopic --sqlEnableNull --createFk --createFkIdx --
   createGeomIdx --createEnumTabs --defaultSrsAuth EPSG --
   defaultSrsCode 2056 --modeldir "http://models.geo.admin.ch;
   datenmodelle/ili2/." --models SO_Nutzungsplanung_20160427 --
   dbschema nplso_2495 --schemaimport
```

Neben den bereits vorher verwendeten Optionen, verwenden wir zusätzlich `--createEnumTabs` und `--createFk`. Mit `--createEnumTabs` werden für die Aufzähltypen Tabellen angelegt, die sich später bei der Erfassung in *QGIS* als sehr praktisch erweisen. Die Option `--createFk` erzeugt Fremdschlüsselbedingungen in Spalten, die Records in anderen Taellen referenzieren.

Weil keine Daten importiert werden müssen, ist der ili2pg-Prozess viel schneller beendet. Falls alles geklappt hat, ist die letzte Meldung auf der Konsole «Info: ...done».

In der Datenbank ist jetzt ein neues Schema «nplso_2495» mit 67 Tabellen vorhanden. In der Tabelle «t_ili2db_model» sind neben dem Nutzungsplanungs-Datenmodell noch fünf weitere INTERLIS-Modelle vorhanden. Diese werden im Nutzungsplanungs-Datenmodell in irgendeiner Art und Weise gebraucht. Die beiden Modelle *GeometryCHLV95_V1* und *CHAdminCodes_V1* werden im Nutzungsplanungs-Datenmodell explizit importiert.

3.3.1 Abbildung von Aufzähltypen

Die Aufzähltypen wurden dank der Option `--createEnumTabs` in separaten Tabellen angelegt und importiert. Das beginnt mit dem Klassiker «halignment» (siehe Abbildung 3.3) und endet bei den Aufzähltypen für die verschiedenen Nutzungstypen (siehe Abbildung 3.4).

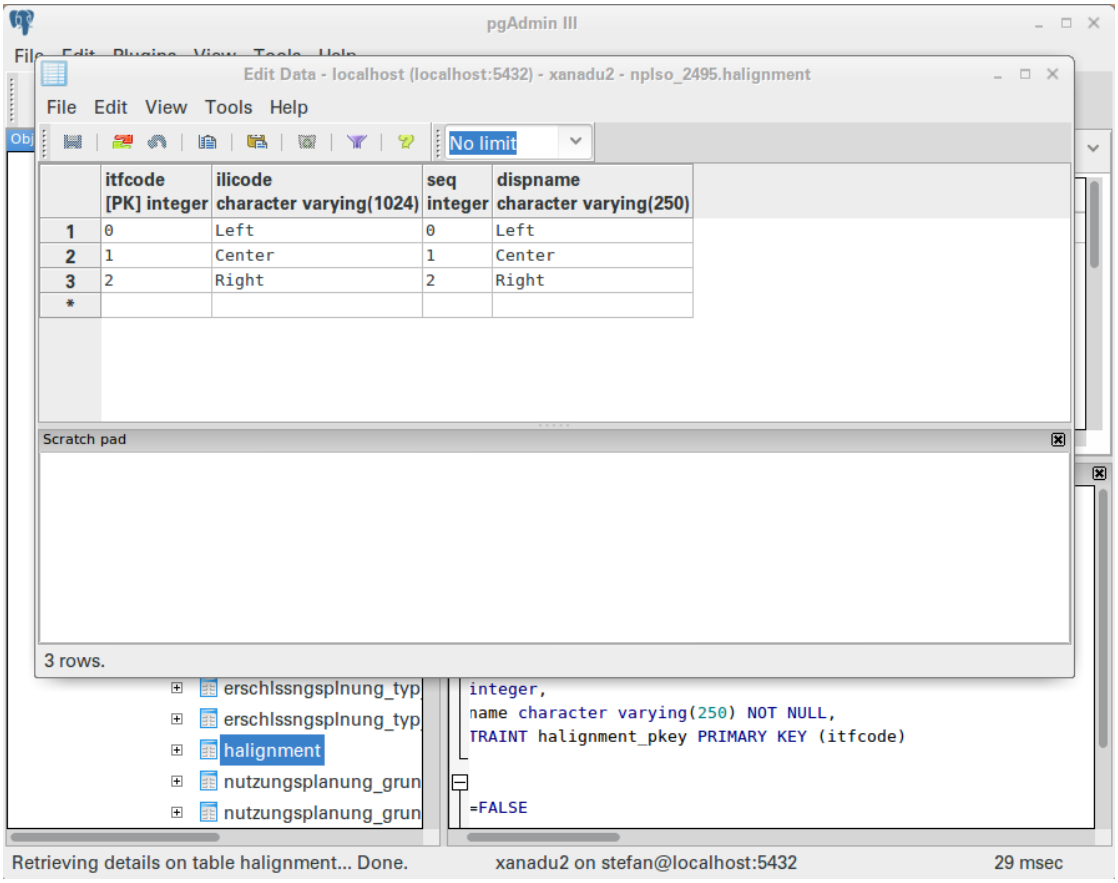


Abbildung 3.3: Aufzähltyp «halignment»

Die Tabelle besteht aus vier Spalten. Für uns wichtig wird nur «ilicode» sein. Sie beinhaltet sämtliche qualifizierte Werte des Aufzählтyps gemäss Datenmodell. Die Spalte «itfcode» enthält den entsprechenden Integer-Wert gemäss INTERLIS 1. Die Spalte «seq» ist leer, da kein geordneter («ORDERED») Aufzählтyp verwendet wird. Die Spalte «dispname» dient dazu anwenderfreundlichere Texte durch den Benutzer selbst anzugeben, z. B. «Strasse / Weg» anstelle von «befestigt.Strasse_Weg» oder «Gebäude» statt «Gebaeude».

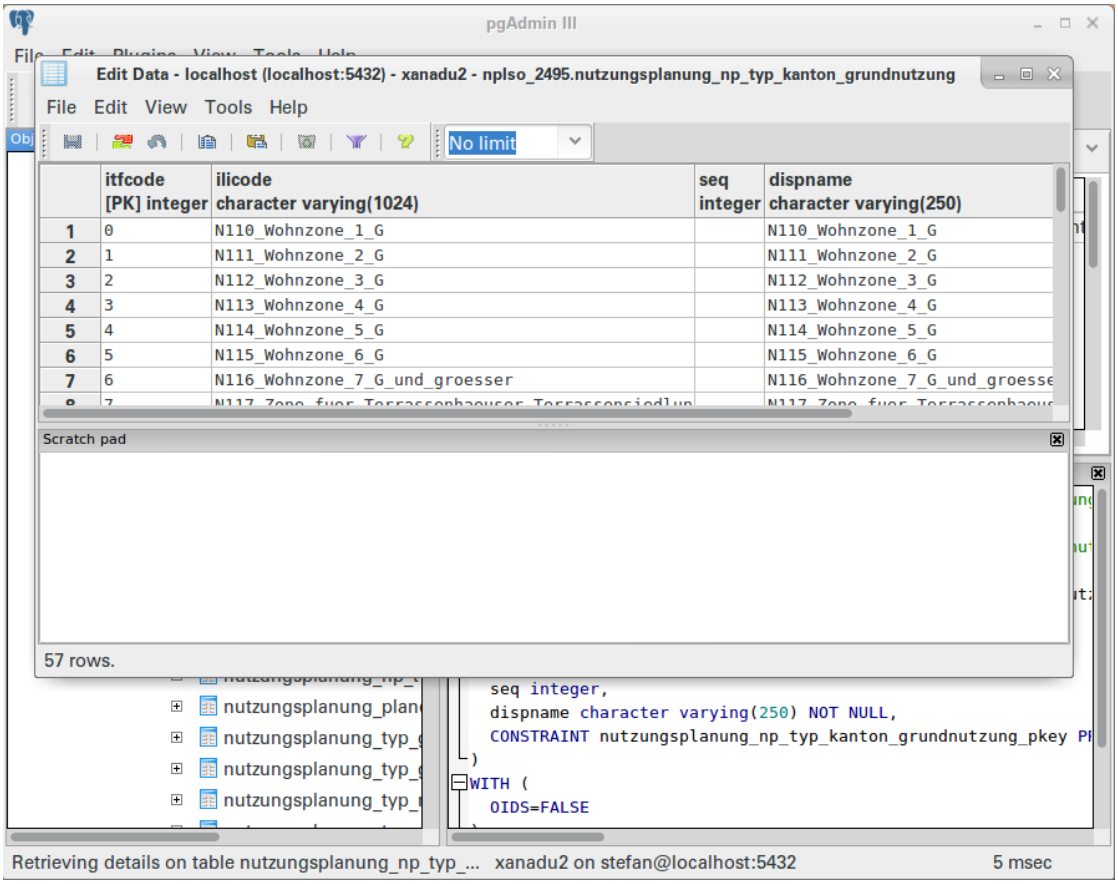


Abbildung 3.4: Aufzählтyp «NP_Typ_Kanton_Grundnutzung»

3.3.2 Abbildung von Vererbungen

Vererbungen können auf ganz unterschiedliche Arten in der Datenbank abgebildet werden. *Ili2pg* verwendet standardmässig eine sogenannte «smarte» Vererbungsstrategie:

«Bildet die Vererbungshierarchie mit einer dynamischen Strategie ab. Für Klassen, die referenziert werden und deren Basisklassen nicht mit einer NewClass-Strategie abgebildet werden, wird die NewClass-Strategie verwendet. Abstrakte Klassen werden mit einer SubClass-Strategie abgebildet. Konkrete Klassen, ohne Basisklasse oder deren direkte Basisklassen mit einer SubClass-Strategie abgebildet werden, werden mit einer NewClass-Strategie abgebildet. Alle anderen Klassen werden mit einer SuperClass-Strategie abgebildet.»[2]

In unserem Fall hat das der positive Effekt, dass der Inhalt einer konkreten Positions-Tabelle (siehe Abbildung 3.2) in der Datenbank nicht auf zwei Tabellen verteilt ist, sondern nur in einer, obwohl die Attribute im Datenmodell nur in der abstrakten Klasse vorkommen und in der konkreten Klasse keine weiteren Attribute dazu kommen:

```
CREATE TABLE nplso_2495.nutzungsplanung_grundnutzung_pos
(
  t_id integer NOT NULL DEFAULT ...,
  aelement integer,
  ori integer,
  hali character varying(255),
  vali character varying(255),
  groesse character varying(255) NOT NULL,
  pos geometry(Point,2056),
  ...
);
```

Hätte man die smarte Vererbungsstrategie ausgeschaltet (z. B. mit `--noSmartMapping`) wären die Informationen in zwei Tabellen verteilt gewesen (NewClass-Strategie) und noch viel wichtiger: Die Erfassung einer Position in *QGIS* müsste in zwei Tabellen stattfinden:

```
CREATE TABLE nplso_2495_nosmart.nutzungsplanung_pos
(
  t_id integer NOT NULL DEFAULT ...,
  t_type character varying(60) NOT NULL,
```

[2] ili2db-Anleitung

```
  ori integer,
  hali character varying(255),
  vali character varying(255),
  groesse character varying(255) NOT NULL,
  pos geometry(Point,2056),
  ...
);

CREATE TABLE nplso_2495_nosmart.
   nutzungsplanung_grundnutzung_pos
(
  t_id integer NOT NULL,
  aelement integer,
  ...
);
```

Als weiteres gutes Anschauungsbeispiel für die smarte Vererbungsstrategie dient die Vererbung «Typ» ← «Typ_Grundnutzung» des Topics «Nutzungsplanung».

3.3.3 Abbildung von Assoziationen

Das gezeigte Beispiel der Vererbung in Kapitel 3.3.2 zeigt auch gerade die Abbildung einer Assoziation (hier in Form einer Aggregation):

```
ASSOCIATION Grundnutzung_Grundnutzung_Pos =
  Element -<> {1} Grundnutzung;
  Pos -- {0..*} Grundnutzung_Pos;
END Grundnutzung_Grundnutzung_Pos;
```

Da es sich dabei um eine 1:mc-Beziehung handelt, steht in in der Tabelle «nutzungsplanung_grundnutzung_pos» neben den Attributen aus dem Datenmodell auch der Fremdschlüssel «aelement» zur Tabelle «nutzungsplanung_grundnutzung». Die Rolle der Aggregation heisst im Modell nur «element», weil es sich dabei anscheinend um

ein reserviertes Wort handelt, setzt *ili2pg* automatisch ein «a»-Prefix vor den Attributnamen. Das spielt keine Rolle, denn *ili2pg* weiss welche Attribute beim Import resp. beim Anlegen der Tabellen umbenannt wurden.

Eine weitere Assoziation wurde in Abbildung 3.1 gezeigt. Dabei handelt es sich um eine n:m-Beziehung (auf sich selbst):

```
ASSOCIATION HinweisWeitereDokumente =
  Ursprung -- {0..*} Dokument;
  Hinweis -- {0..*} Dokument;
END HinweisWeitereDokumente;
```

Wie bildet *ili2pg* diese Assoziation in der Datenbank ab?

```
CREATE TABLE nplso_2495.
   rechtsvorschrften_hinweisweiteredokumente
(
  t_id integer NOT NULL DEFAULT ...,
  ursprung integer NOT NULL,
  hinweis integer NOT NULL,
  ...
);
```

Es entsteht eine neue Tabelle (cross-reference table) in der Datenbank. Diese Tabelle referenziert die jeweiligen Primärschlüssel der Tabelle «rechtsvorschrften_dokument».

3.3.4 Abbildung von Strukturen

Das Modell verwendet keine Strukturen. Um aber einen Überblick zu bekommen, wie *ili2pg* Strukturen in der Datenbank abbildet, soll das Thema hier kurz theoretisch abgehandelt werden.

Im Allgemeinen werden Strukturen wie Klassen abgebildet, d. h. jede Struktur wird in eine Datenbanktabelle abgebildet. Für jedes Strukturattribut wird in dieser Tabelle eine

zusätzliche Spalte für den Fremdschlüssel erstellt. Die Tabelle enthält auch zusätzlich eine Spalte «T_seq», welche die Reihenfolge der Strukturelemente festlegt (bei «LIST OF»).

In der ili2db-Anleitung wird die Theorie mit einem Beispiel in INTERLIS und SQL erläutert.

3.3.5 Abbildung von Multi-Geometrien

INTERLIS kennt bis und mit Version 2.3 keinen nativen Multi-Geometrie-Datentyp. Mit dem Aufkommen der CHBase-INTERLIS-Modellen hat man angefangen sich einen MultiSurface-Geometrie-Typ mittels Strukturen und «BAG OF»Konstrukten zusammen zu basteln. Das funktioniert soweit gut und ist auch zulässig:

```
STRUCTURE SurfaceStructure =
  Surface: Surface;
END SurfaceStructure;

STRUCTURE MultiSurface =
  Surfaces: BAG {1..*} OF SurfaceStructure;
END MultiSurface;
```

Ili2pg kann mit dieser Konstruktion problemlos umgehen. Der Nachteil ist aber, dass die ganze Struktur-Geschichte (siehe Kapitel 3.3.4) in diesem Fall in der Datenbank für die effiziente Datenerfassung mit *QGIS* nicht mehr sinnvoll abgebildet wird. Störend ist, dass das einzelne Polygon (nicht Multipolygon) getrennt von den Sachattributen in einer anderen Tabelle verwaltet wird.

Ili2pg wurde dahingehend erweitert, dass beim Verwenden von CHBase-Multi-Geometrie-Typen direkt Multipolygone in der gleichen Tabelle wie die Sachattribute abgebildet werden. Somit wird auch die Datenerfassung massiv vereinfacht.

Leider funktioniert das wirklich nur mit den CHBase-Geometrie-Typen und nicht, wenn man diese Geometrie-Typen in einem eigenen Modell «nachbaut». Mit INTERLIS 2.4 werden mit grosser Wahrscheinlichkeit aber Multi-Geometrien generisch unterstützt.

3.4 Fazit

Weniger wichtig *wie* das O/R-Mapping (also welche Variante) gemacht wird, sondern viel wichtiger ist, dass es eine Schnittstellensoftware für INTERLIS gibt, die das macht und man *versteht*, wie sie es macht.

4 Datenerfassung mit QGIS

4.1 Ziele

- Benutzerdefinierte Eingabeformulare in *QGIS* kennenlernen:
  - Bearbeitungselemente (Edit widgets)
  - Beziehungen (Relations)
- Daten direkt in einem Datenmodell erfassen

4.2 Bestehender Zonenplan

Ist der bestehende Zonenplan z. B. als PDF vorhanden, kann man das PDF in *QGIS* georeferenzieren. Die georeferenzierte Rasterdatei[1] dient uns zusammen mit der amtlichen Vermessung als Datengrundlage für das Erfassen der Nutzungsplanung.

4.3 Bearbeitungselemente

Standardmässig sind in *QGIS* die Eingabeformulare sehr einfach gehalten. Für jedes Attribut erscheint ein einfaches Textfeld (siehe Abbildung 4.1). Je nach Datentyp können nur Zahlen eingegeben werden.

[1] daten/nplso/nied_pilotprojekt_nup_151217.tif

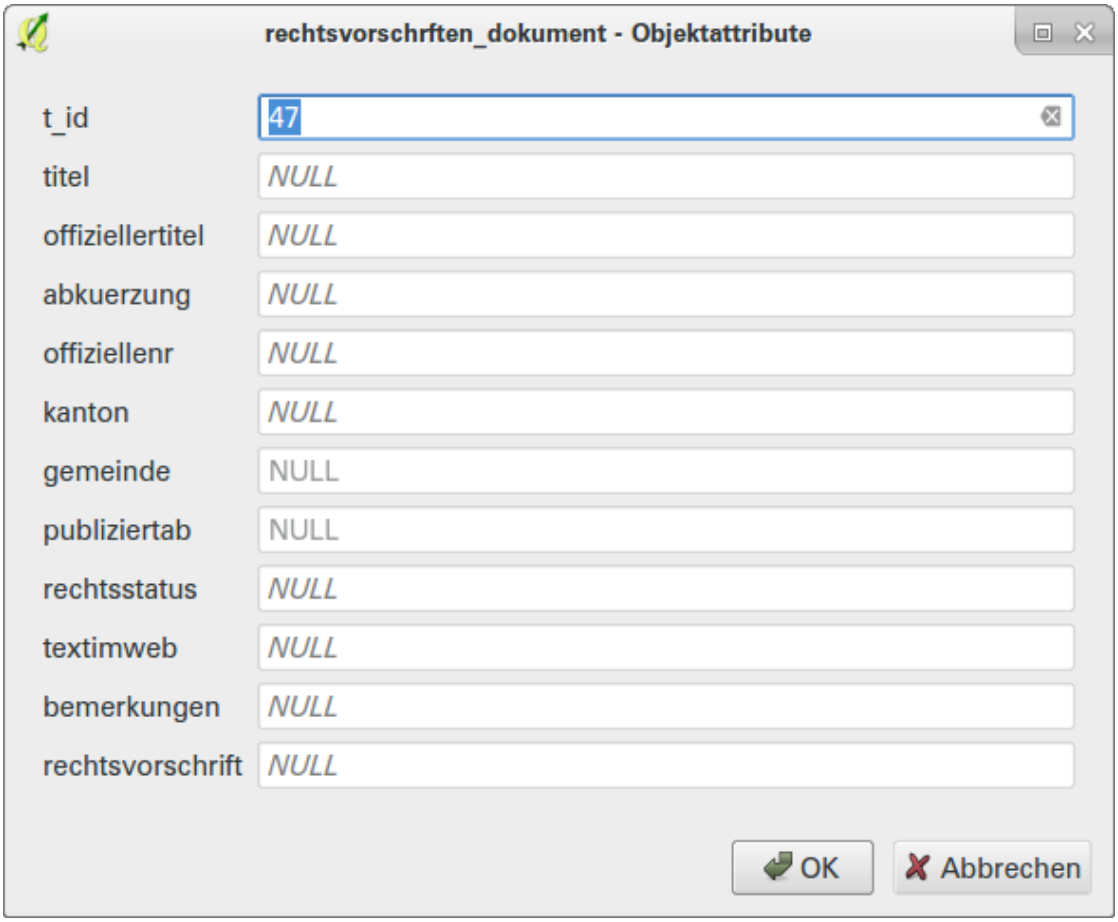


Abbildung 4.1: Formular mit Standardeinstellung

Die Eingabemasken können für Objekte mit oder ohne geografischen Bezug individuell angepasst werden. Dank den Anpassungen kann die Effizienz bei der Erfassung und die Qualität der Daten selbst gesteigert werden. Sämtliche Einstellungen dazu finden sich unter **Layer > Eigenschaften > Felder**:

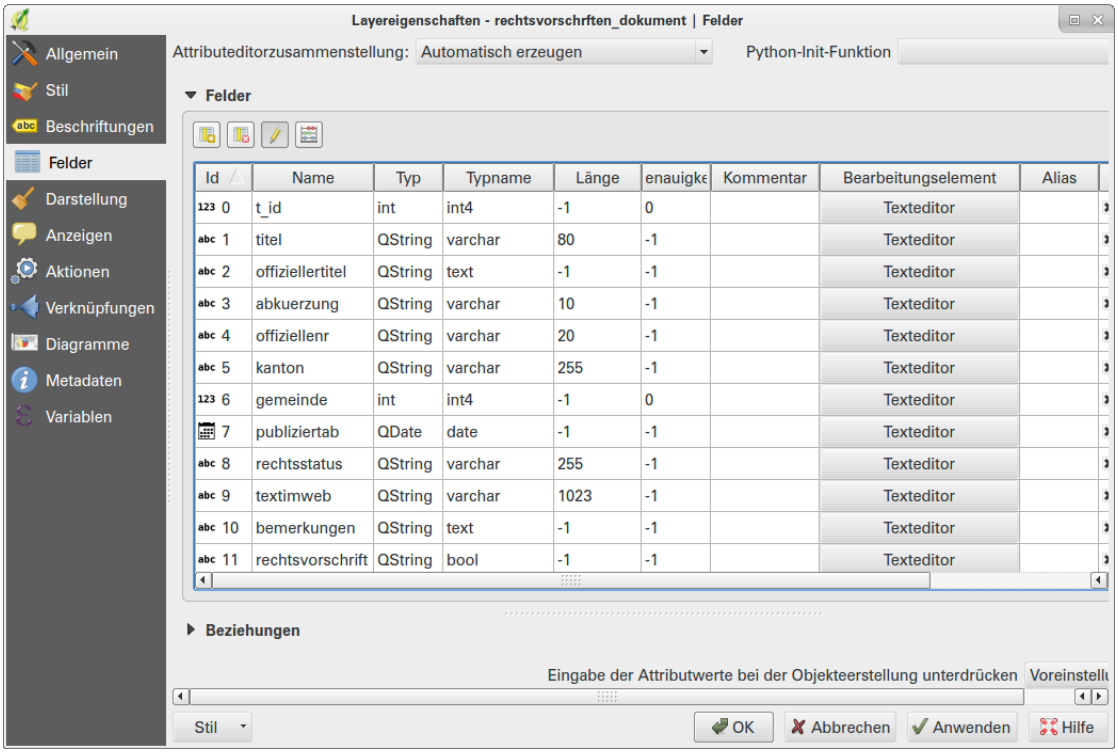


Abbildung 4.2: Formulareigenschaften

Die erste Wahl, die man treffen kann, ist das Aussehen des Formulars resp. wie die Attribute im Formular zusammengestellt werden sollen. Unter **Attributeditorzusammenstellung** gibt es drei Auswahlmöglichkeiten:

1. «Automatisch erzeugen»: Die Attribute werden unabhängig vom gewählten Widget einfach untereinander aufgereiht (siehe Abbildung 4.1).
2. «Mit Drag and Drop zusammenstellen»: Die Attribute können in Tabs und Groupboxen gruppiert werden.
3. «UI-Datei verwenden»: Dabei handelt es sich um die flexibelste Variante. Man kann im Qt-Designer das Formular selber zusammenbasteln.

Im Workshop verwenden wir nur die Varianten 1 und 2.

Unter **Bearbeitungselement** lässt sich das gewünschte Bearbeitungselement (Edit widget) wählen. Es stehen insgesamt 17 Typen mit jeweils vielen Optionen zur Aus-

wahl. Einige lassen sich jedoch nur für bestimmte Datentypen oder -provider auswählen. Die Typen werden in der QGIS-Dokumentation eingehend[2] erläutert.

Exemplarisch wählen wir nun für die Attribute des Layers «rechtsvorschrften_dokument» das passende Bearbeitungselement und setzen wo nötig passende Aliase:

t_id Da ändern wir den Typ nicht, sondern machen das Attribut nur nicht-änderbar. Es handelt sich dabei um den Primärschlüssen, der automatisch durch die Datenbank mit einer Sequenz kontrolliert wird.

titel Das Bearbeitungsfeld wird nicht geändert.

offiziellertitel Das Bearbeitungsfeld wird nicht geändert.

offiziellenr Das Bearbeitungsfeld wird nicht geändert[3].

kanton Jetzt wird es erstmals interessant. Gemäss INTERLIS-Modell wird der Aufzähltyp «CHCantonCode» aus einem CHBase-Modell verwendet. Für Aufzähltypen eignet sich in *QGIS* das Bearbeitungselement «Wertbeziehung». Dazu muss aber der Aufzähltyp-Layer («chcantoncode) zuerst in *QGIS* geladen werden. Es lohnt sich diese Aufzähltyp-Layer in einer separaten Gruppe in *QGIS* zu verwalten (z. B. «Aufzählungen»). Als **Schlüsselspalte** und **Wertespalte** wähle ich jeweils die Spalte «ilicode» aus.

Ist das Attribut gemäss Modell optional, kann man noch **Allow NULL value** anwählen. **Nach Wert sortieren** sortiert die Werte alphanumerisch.

[2] http://docs.qgis.org/2.8/de/docs/user_manual/working_with_vector/vector_properties.html#fields-menu

[3] Gemäss NPLSO-Handbuch v0.3: «vorgegebene Dokumentennummer (korrespondierend mit Plan_Nummer)»

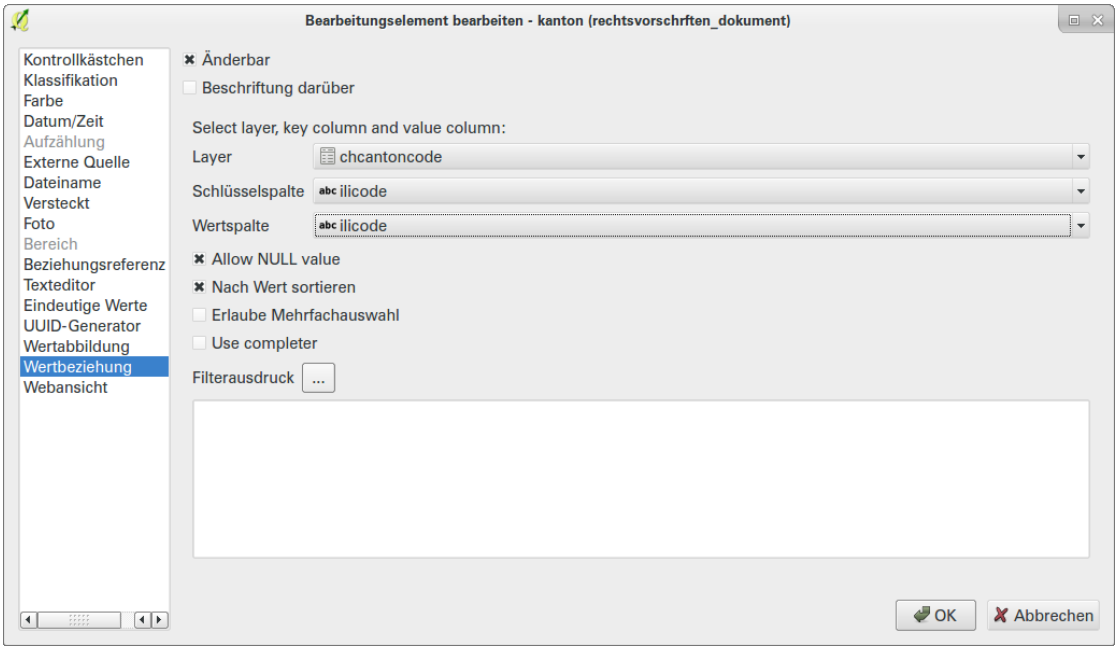


Abbildung 4.3: Bearbeitungsselement «Wertbeziehung»

Das Resultat ist eine Combobox, die sämtliche Werte aus dem Layer «chcantoncode» enthält:

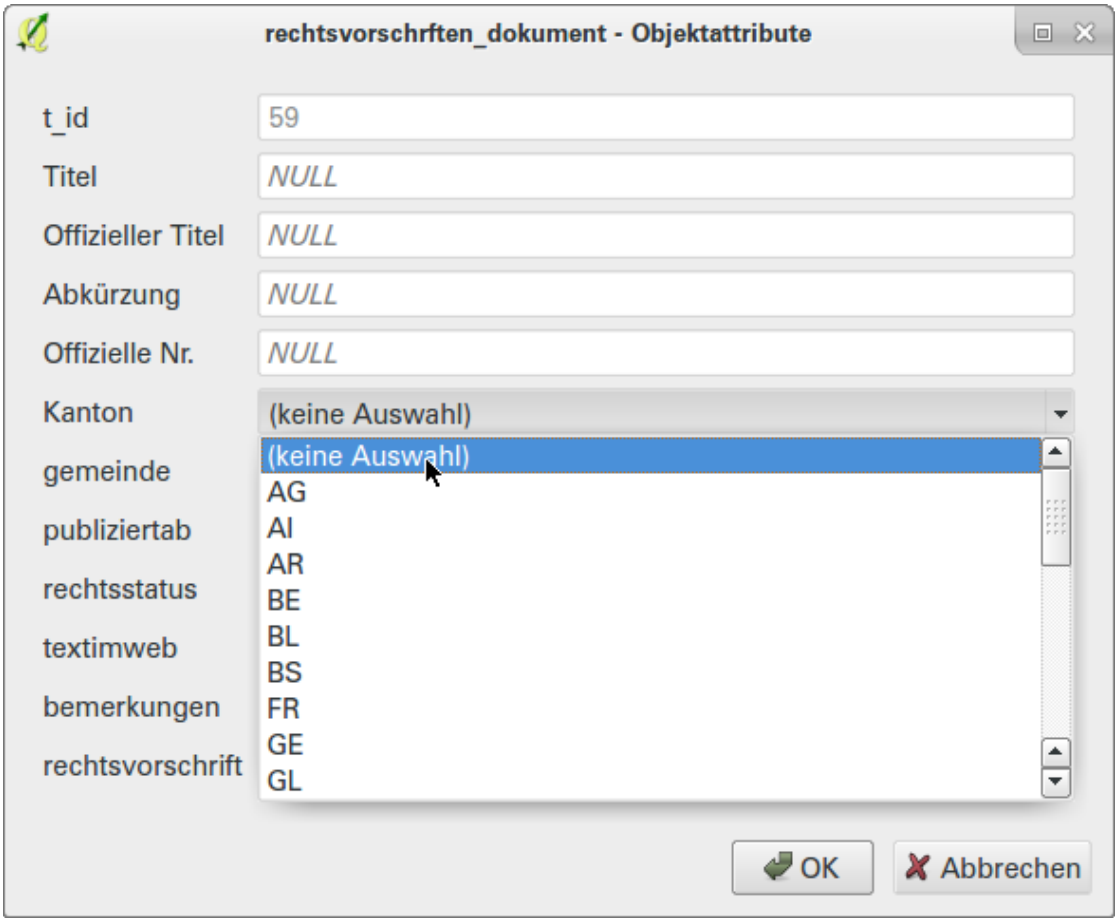


Abbildung 4.4: Wertbeziehung als Combobox

Will man anstelle einer Comobox lieber eine Autovervollständigung kann man die Option **Use completer** wählen. Mit dieser Option erscheint ein einfaches Textfeld und *QGIS* macht nach dem ersten Tippen bereits passende Vorschläge:

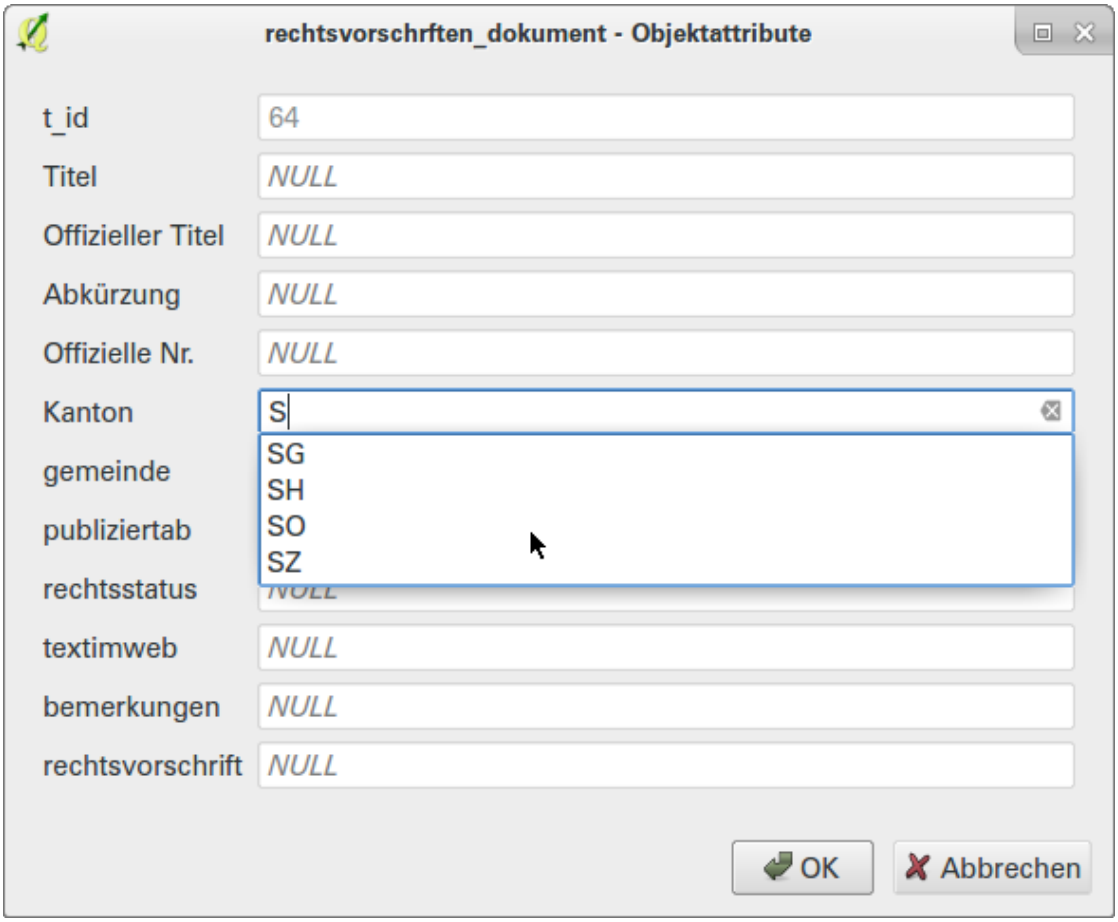


Abbildung 4.5: Wertbeziehung mit «Use completer»-Option

Es gibt noch eine weitere Variante. Weiss man, dass man in dieses Feld immer «SO» tippen wird, kann man in der Datenbank einen Standardwert setzen und das Feld in *QGIS* ausschalten.

gemeinde Beim Attribut «Gemeinde» handelt es sich um die BFS-Gemeindenummer. Im Kanton Solothurn sind das Werte im Bereich von 2401 bis 2622. Für solche Wertebereiche wählt man das Bearbeitungselement «Bereich». Es gibt drei Varianten des eigentlichen Eingabefeldes:

- Änderbar: Einfaches Textfeld mit Hoch- und Runter-Pfeilen.
- Schieber

- Drehregler

Für diesen Anwendungsfall eignet sich «Änderbar». Anschliessend wählt man das Minimum und Maximum und die Schrittweite aus:

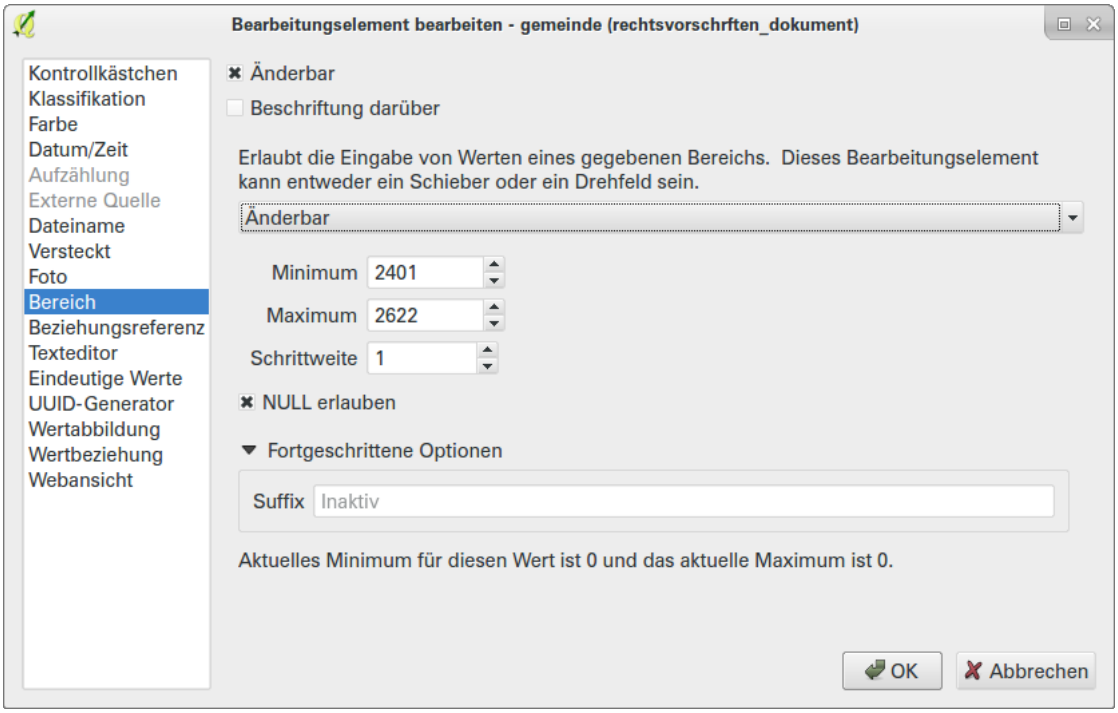


Abbildung 4.6: Wertebereich

publiziertab Für Datumstypen (siehe Abbildung 4.7) verwendet man das Bearbeitungselement «Datum/Zeit». Die Standardeinstellung reichen für unseren Zweck aus. Einzig die Option **Calender popup** sollte man – zwecks einer bequemeren Eingabe – noch anwählen.

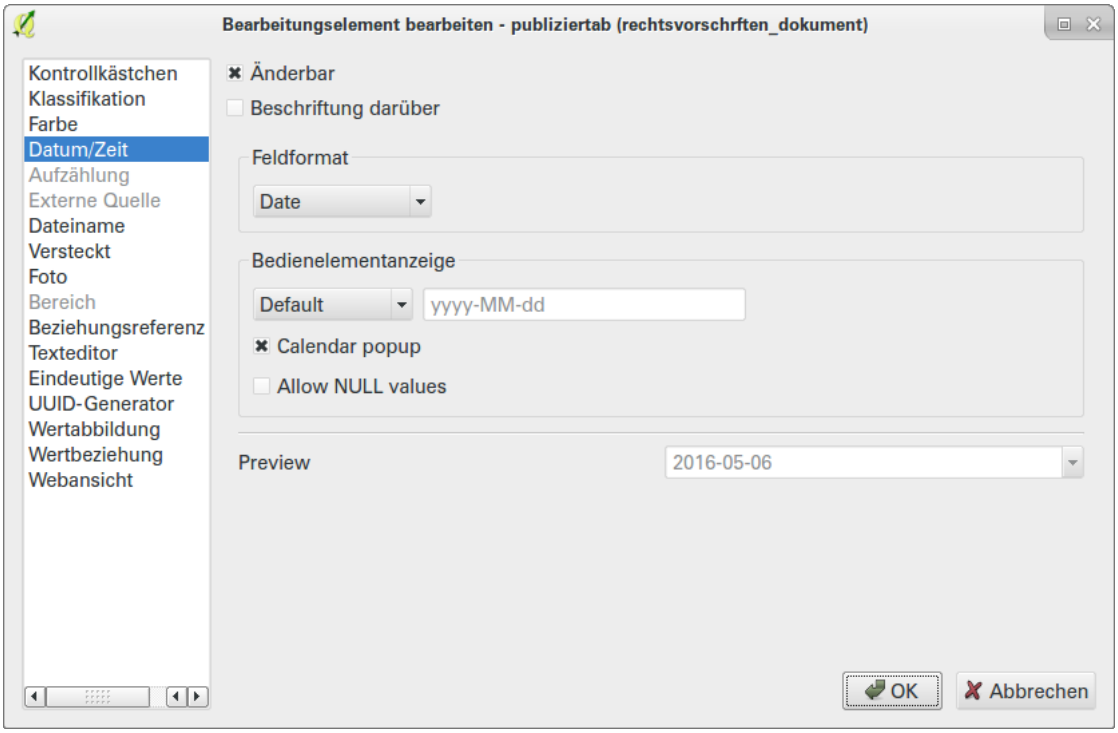


Abbildung 4.7: «Datum/Zeit»-Bearbeitungselement

rechtsstatus Das Attribut «rechtsstatus» ist ein Aufzähltyp. Das Vorgehen ist analog dem Attribut «kanton». Der Aufzähltyp-Layer «rechtsstatus» muss vorgängig in *QGIS* geladen werden.

textimweb Dabei handelt es sich um eine URL zu einem Dokument oder Text im Internet. Dazu wäre das Bearbeitungselement «Webansicht» prädistiniert. In diesem Widget kann man einen Dateinamen oder eine URL zu einer Webseite angeben und *QGIS* stellt den Inhalt im Formular dar. Leider scheint es mit PDF-Dateien nicht klarzukommen und aus diesem Grund belassen wir es beim Bearbeitungselementtyp «Texteditor».

bemerkungen Das letzte Attribut ist vom Typ «BOOLEAN». Dazu passend gibt es das Bearbeitungselement «Kontrollkästchen». Als Option muss man einzig angeben, welchen Wert für das angehakte Kästchen resp. nicht angehakte Kästchen in die Datenbank geschrieben wird. Im Fall von *PostgreSQL* ist es «t» resp. «f».

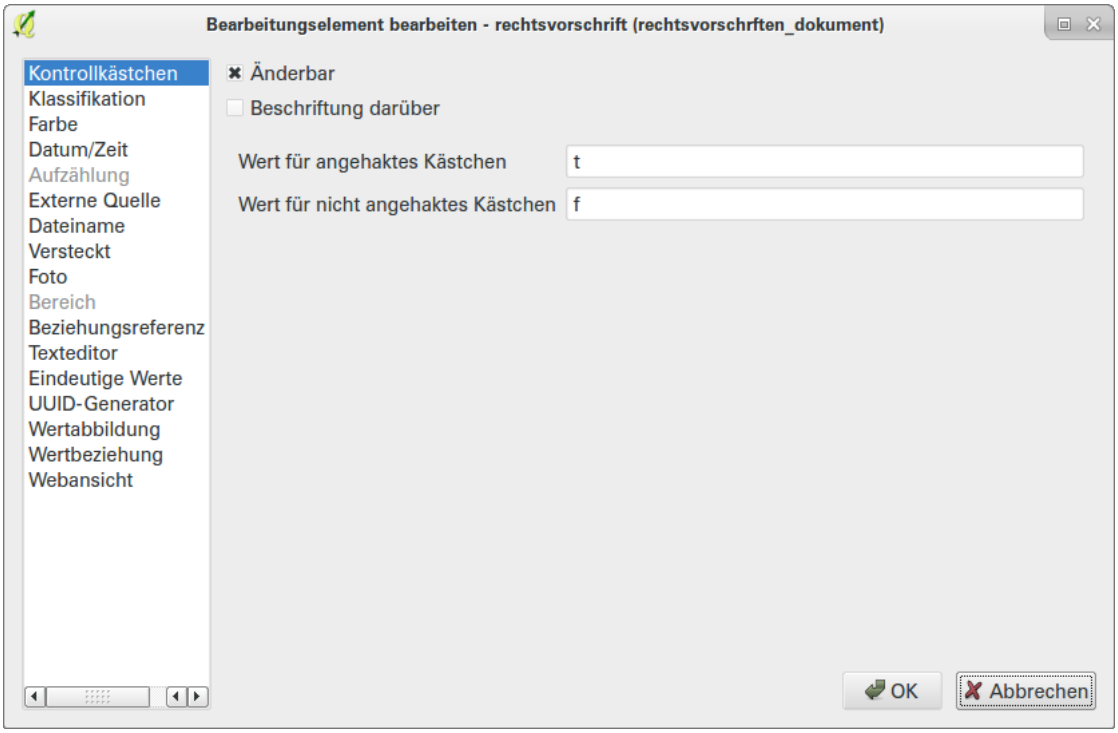


Abbildung 4.8: Kontrollkästchen für «boolean»-Werte

Das Endresultat (Abbildung 4.9) sieht dann so aus:

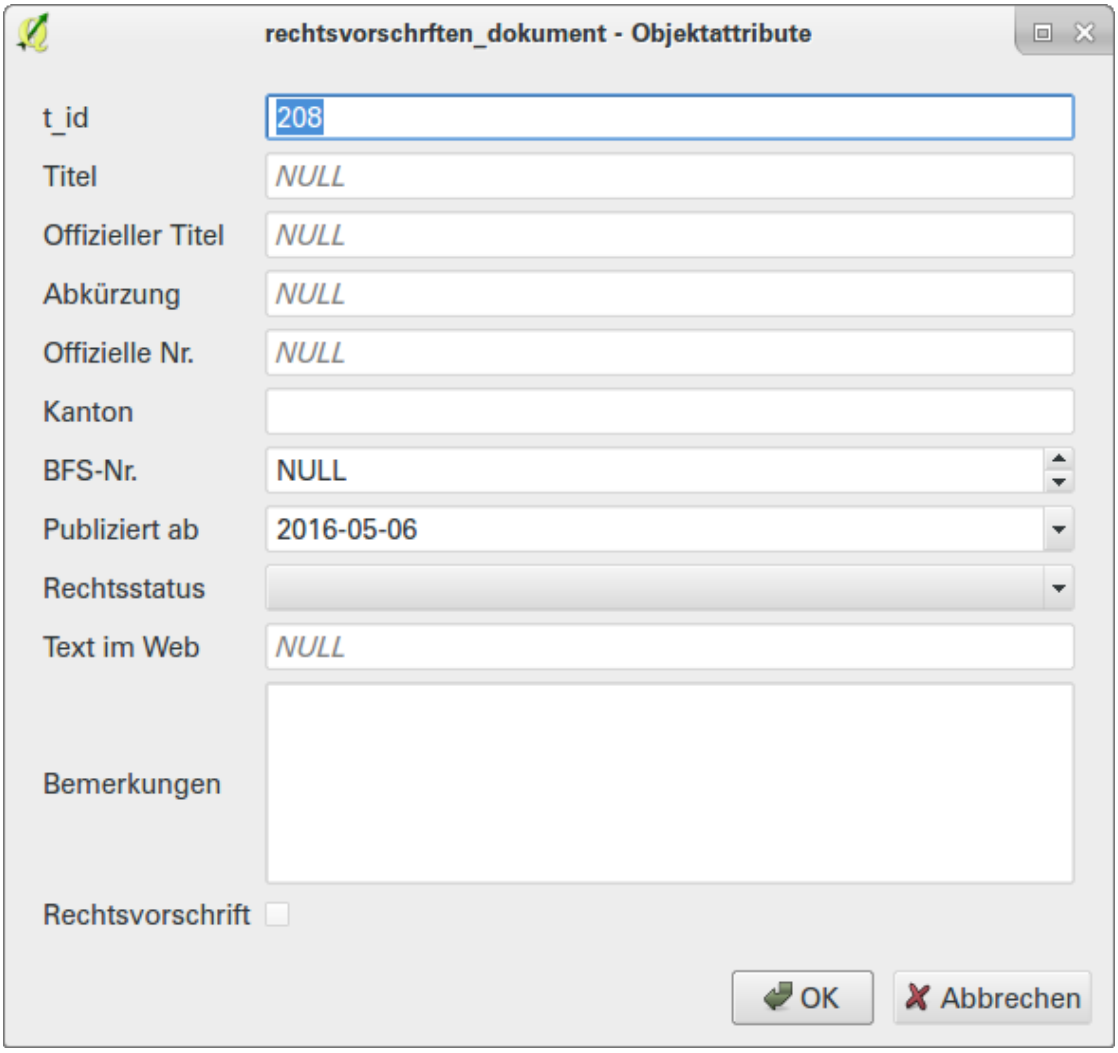


Abbildung 4.9: Individualisierte Eingabemaske

Um die getane Arbeit auszutesten, können wir nun ein Feature erfassen:

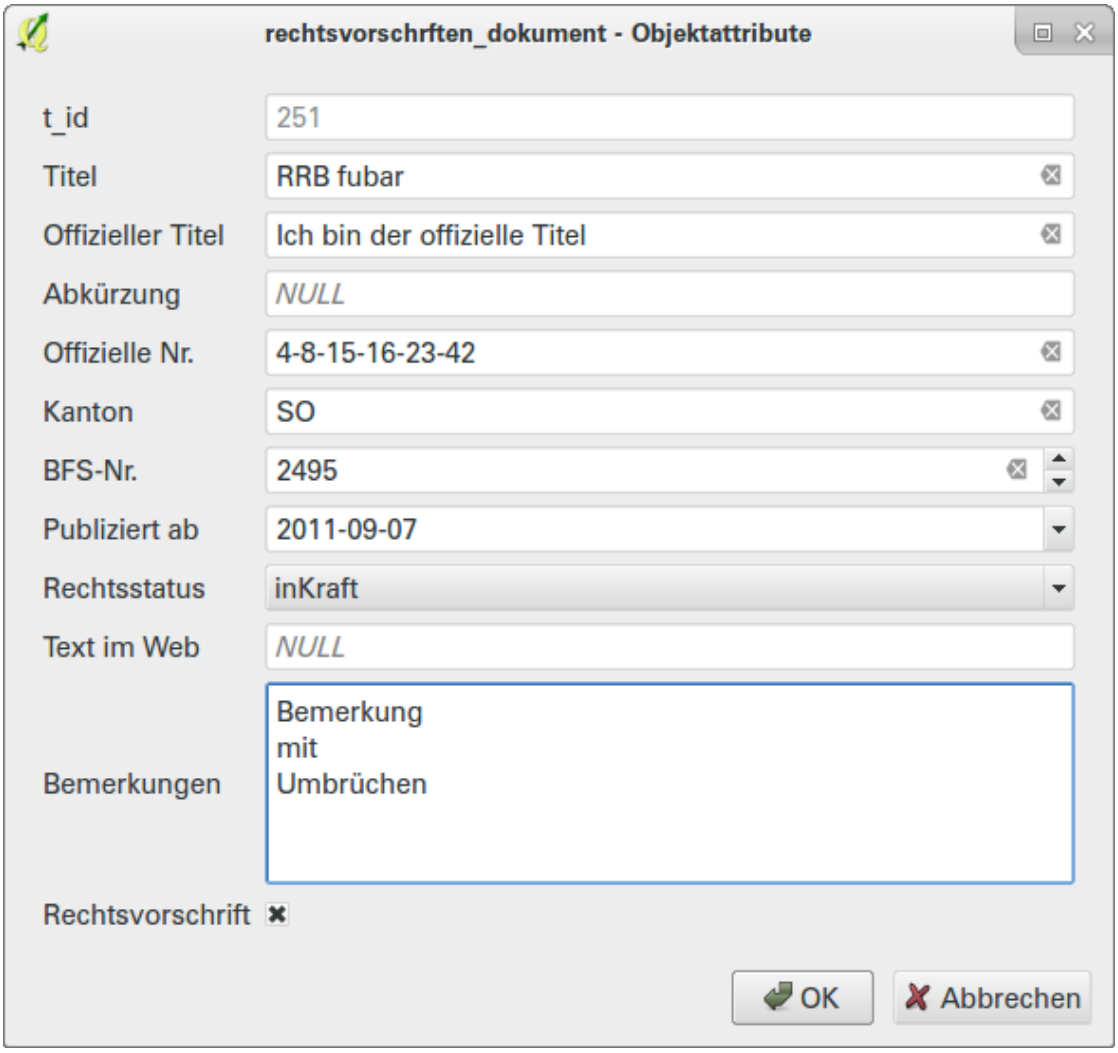


Abbildung 4.10: Testerfassung

Wurde in allen Pflichtfeldern etwas eingetragen, kann das Feature in der Datenbank gespeichert werden. Falls ein Pflichtfeld fehlt, erscheint in *QGIS* eine Fehlermeldung wegen Constraint-Verletzungen («violates not-null constraint»).

In *pgAdmin3* können wir überprüfen, ob es tatsächlich gespeichert wurde:

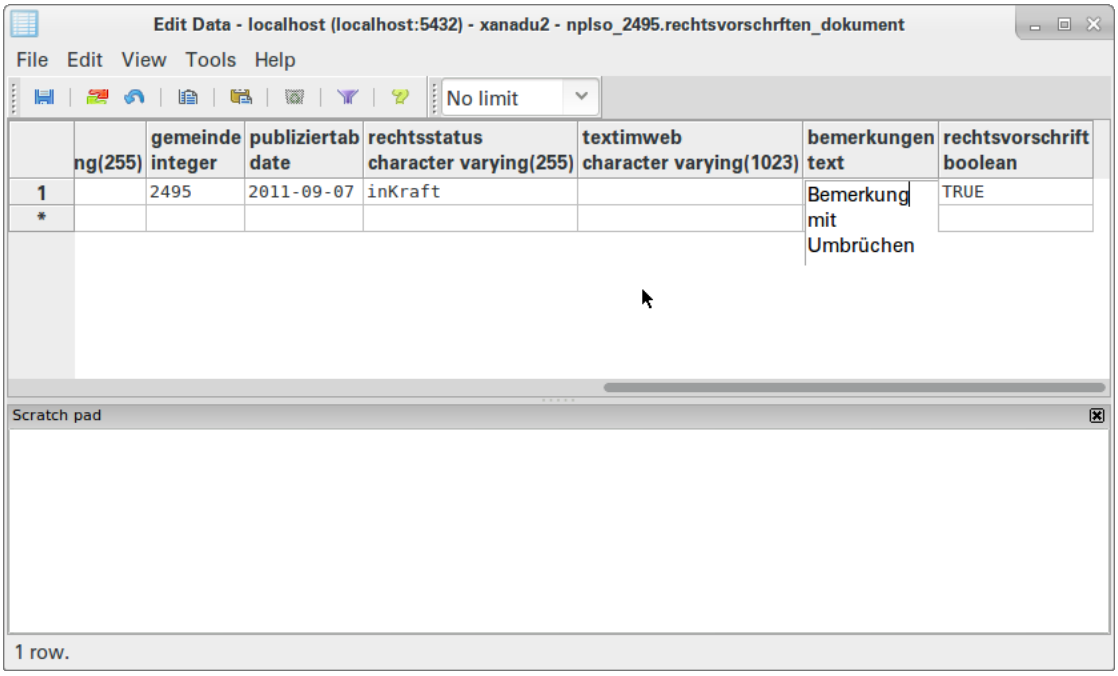


Abbildung 4.11: Testobjekt in pgadmin3

4.4 Beziehungen

In einer relationalen Datenbank dreht sich – nomen est omen – vieles um Beziehungen. Die vielen Relationen aus dem Nutzungsplanung-Datenmodell wurden alle in irgendeiner Form in der Datenbank abgebildet. *QGIS* bietet mit den Beziehungs-Funktionen (Relations) eine fantastische Möglichkeit die Datenerfassung massiv zu vereinfachen oder sogar erst sinnvoll zu ermöglichen.

Als erstes Beispiel herhalten muss wiederum die Assoziation «HinweisWeitereDokumente» aus Abbildung 3.1:

```
ASSOCIATION HinweisWeitereDokumente =
  Ursprung -- {0..*} Dokument;
  Hinweis -- {0..*} Dokument;
END HinweisWeitereDokumente;
```

Diese n:m-Beziehung auf sich selbst wird mit einer zusätzlichen Tabelle «rechtsvorschrften_hinweisweiteredokumente» in der Datenbank abgebildet. Diese referenziert die jeweiligen Primärschlüssel der Tabelle «rechtsvorschrften_dokument». Man könnten jetzt einfach die Tabelle «rechtsvorschrften_hinweisweiteredokumente» in *QGIS* laden und die beiden Attribute «ursprung» und «hinweis» jeweils manuell erfassen.

Dies wird auf Dauer mühsam und ist garantiert auch fehleranfällig, da man sich nichtssagende Primärschlüssel merken muss. Mit den Beziehungs-Funktionen von *QGIS* geht das jetzt schöner, einfacher und sicherer. Die Beziehungs-Funktionen sind etwas Ähnliches wie das Bearbeitungselement «Wertbeziehung» nur viel mächtiger. Insbesondere weil man beide Seiten der Beziehung anschauen und bearbeiten kann. Die Tabelle «rechtsvorschrften_hinweisweiteredokumente» löst die ursprüngliche n:m-Beziehung aus dem INTERLIS-Datenmodell in zwei 1:m-Beziehungen in der Datenbank auf. In unserem (eher) Spezialfall sind nicht drei Tabellen[4] involviert, sondern bloss zwei.

Die Relations in *QGIS* muss man unter **Projekt > Projekteigenschaften > Beziehungen** erstellen. Als «Name» kann man wählen, was man will. Es lohnt sich aber einem gewissen Schema zu folgen. Anschliessend muss man jeweils für beide Enden der Beziehung den Layernamen und die beiden referenzierenden Attribute auswählen. Oder in QGIS-Sprech: das referenz*ierende* Feld und das referenz*ierte* Feld. Die beste Hilfestellung sind die Klammern: Child und Parent. Die «Id» der Beziehung kann man getrost automatisch erzeugen lassen:

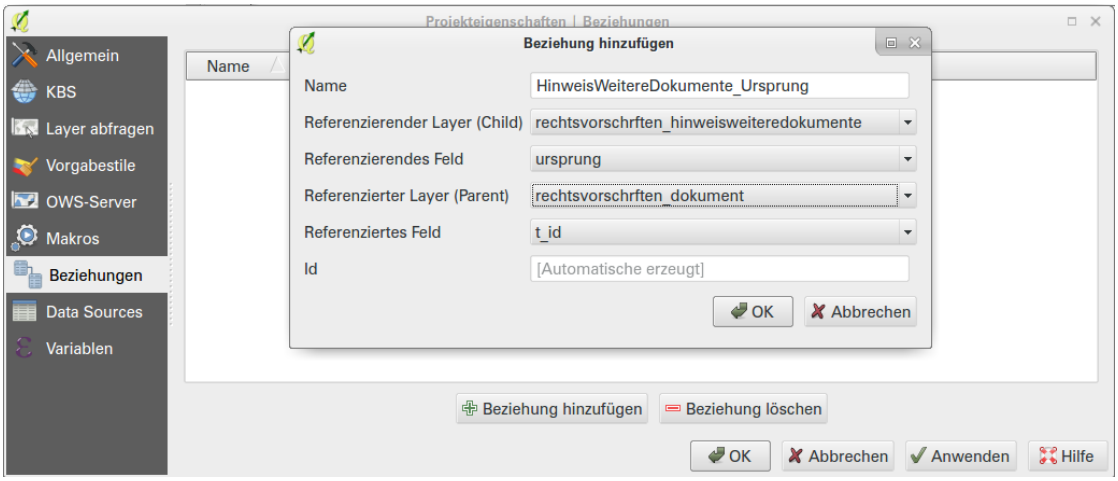


Abbildung 4.12: 1:m-Beziehung

[4] Klassiker: Vorlesung - Student

Wenn wir beide Beziehung erstellt haben, fällt beim Öffnen der Attributtabellen in der Formularansicht sofort auf, dass etwa Neues am unteren Rand hinzugekommen ist. Es sind die beiden Relationen:

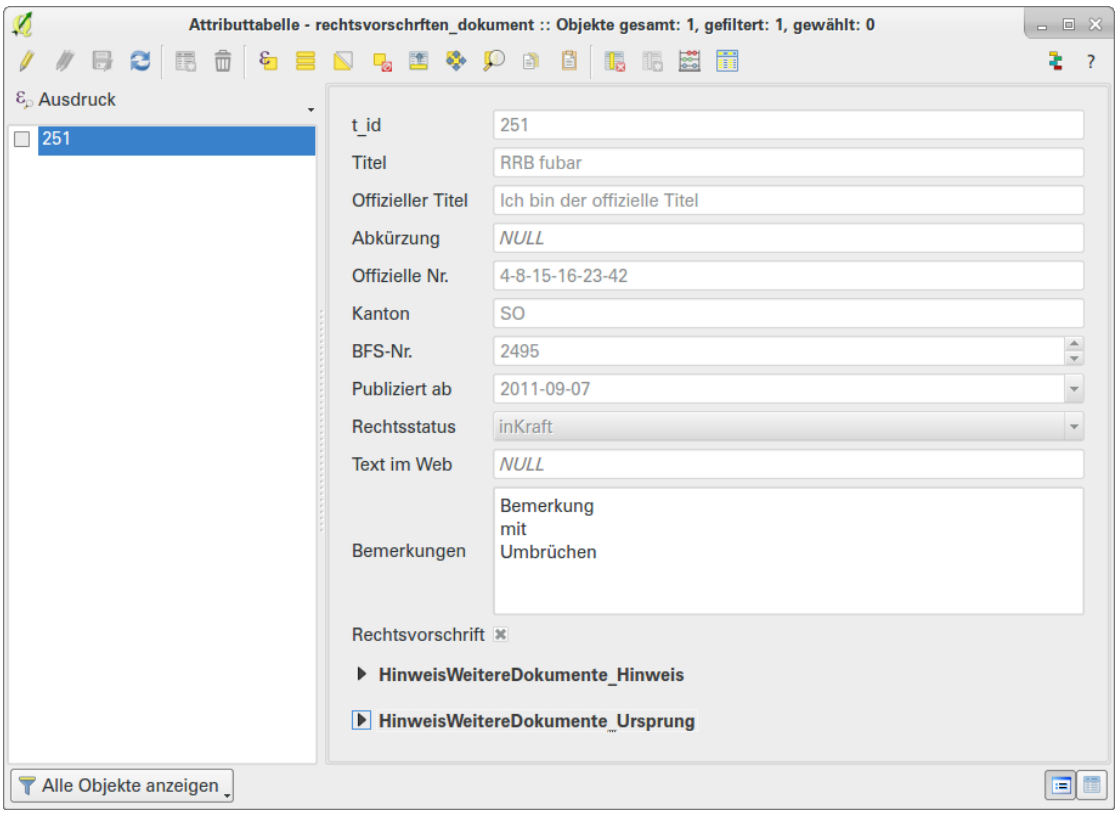


Abbildung 4.13: Relationen in der Formularansicht

Diese beiden Relationen zeigen nun auf die referenzierten Objekte der Tabellen «rechtsvorschrften_hinweisweiteredokumente». Man kann aber auch von hier aus in dieser Tabelle Daten erfassen, die nicht mit dem angewählten Feature verknüpft sind.

In dieser Form sieht das ziemlich unübersichtlich aus. Je nach Arbeitsweise, d. h. wie, wie oft, eventuell auch zu welchem Zeitpunkt die Daten oder die Beziehung zwischen erfasst werden, lohnt es sich die beiden Relationen in einen separaten Tab auszulagern. Dazu wechseln wir in **Layer > Eigenschaften > Felder** bei **Attributeditorzusammenstellung** von «Automatisch erzeugen» zu «Mit Drag and Drop zusammenstellen» (siehe Abbildung 4.14). Es sind zwei Reiter «Daten» und «Beziehungen» zu erstellen. Im «Daten»-Reiter verwalten wir alle Attribute und im «Beziehungen»-Reiter sämtliche

Beziehungen. Sind die Beziehungen in diesem Eingabeformular nie von Interesse, können sie auch weggelassen werden.

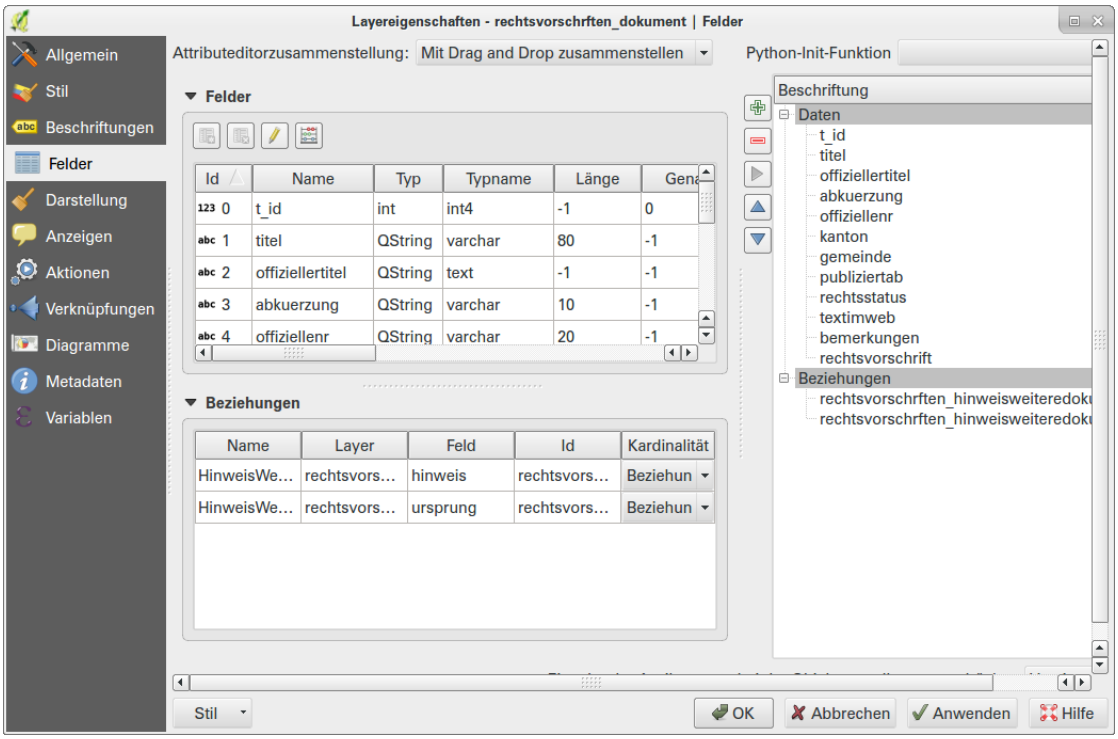


Abbildung 4.14: Eingabeformular «Mit Drag and Drop zusammenstellen»

Das Eingabeformular sieht neu bereits viel aufgeräumter aus:

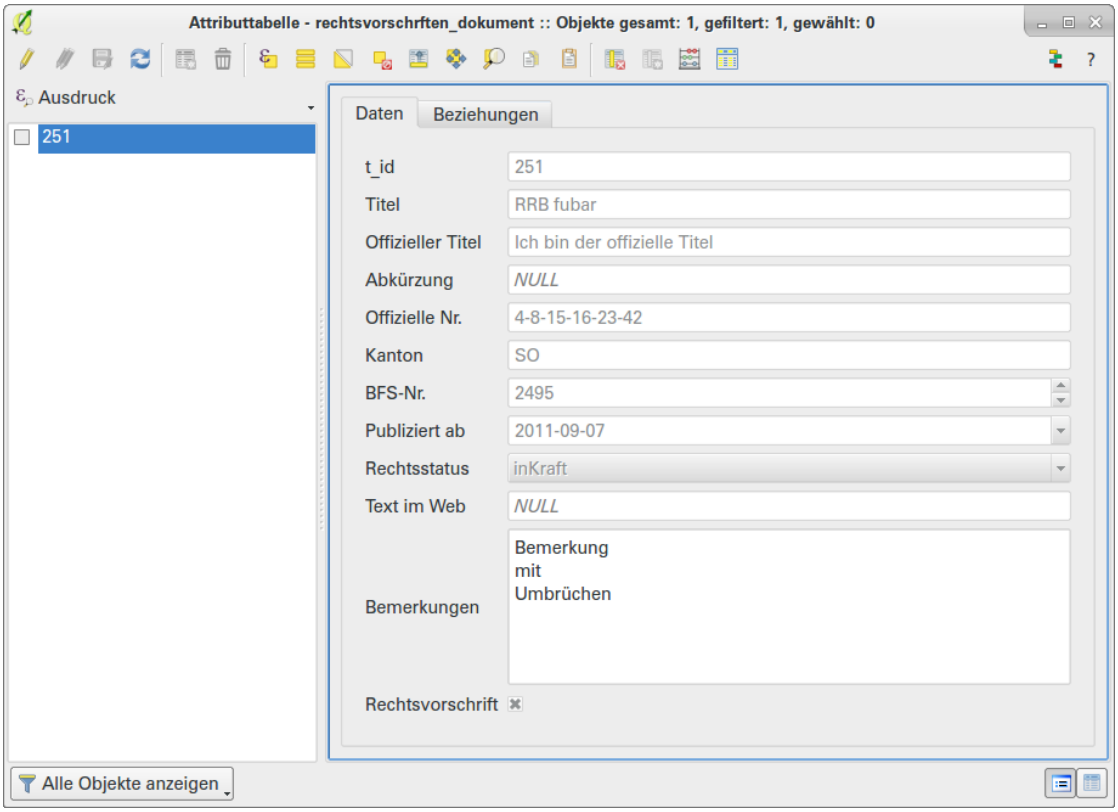


Abbildung 4.15: Eingabeformular mit Reiter

Um überhaupt sinnvoll eine Beziehung erfassen zu können, müssen wir ein weiteres «Dokument» (also den Hinweis zum Ursprung) erfassen[5]:

[5] Das doppelte Kodierungszeichen scheint in INTERLIS 2 erlaubt zu sein (→ Anhang B des Referenzhandbuches).

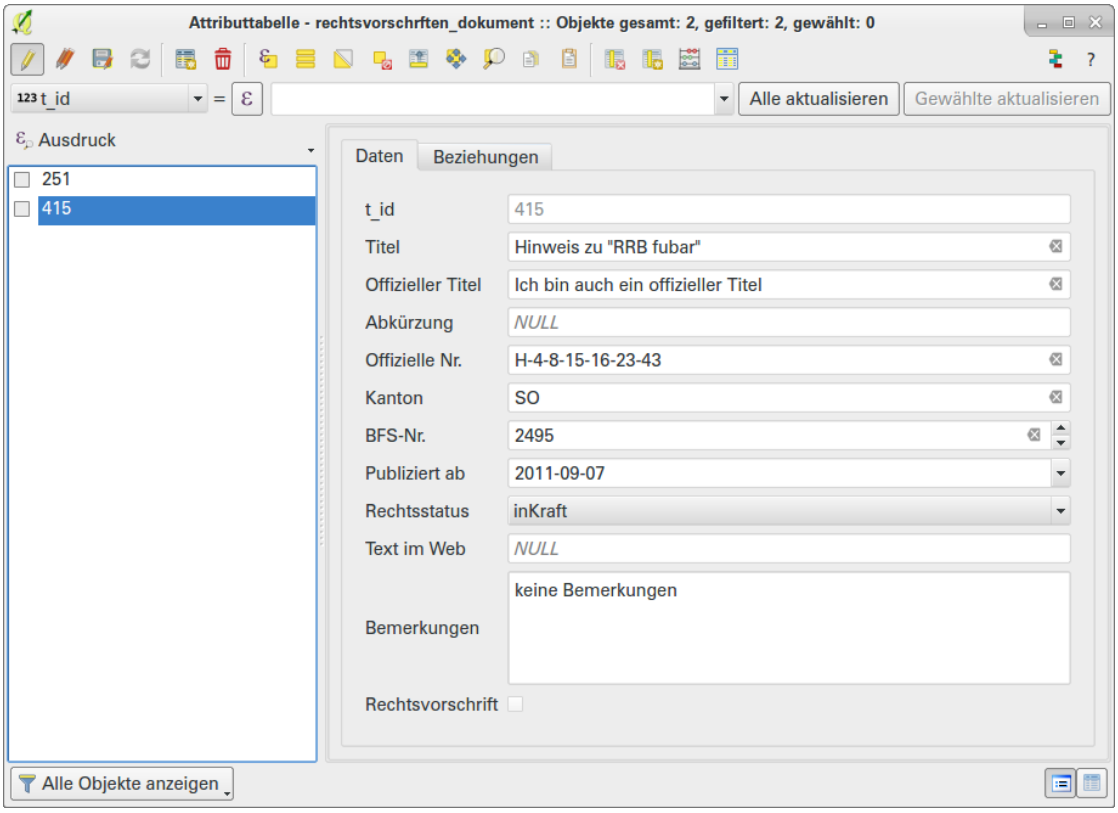


Abbildung 4.16: Zweites «Dokument» (Hinweis zu Ursprung)

Wenn man jetzt die Beziehung zwischen den beiden Dokumenten in der Tabelle «rechtsvorschrften_hinweisweiteredokumente» herstellen will, in dem man das referenzierte und referenzierende Feld erfasst, sieht das noch nicht viel bequemer aus als vorher:

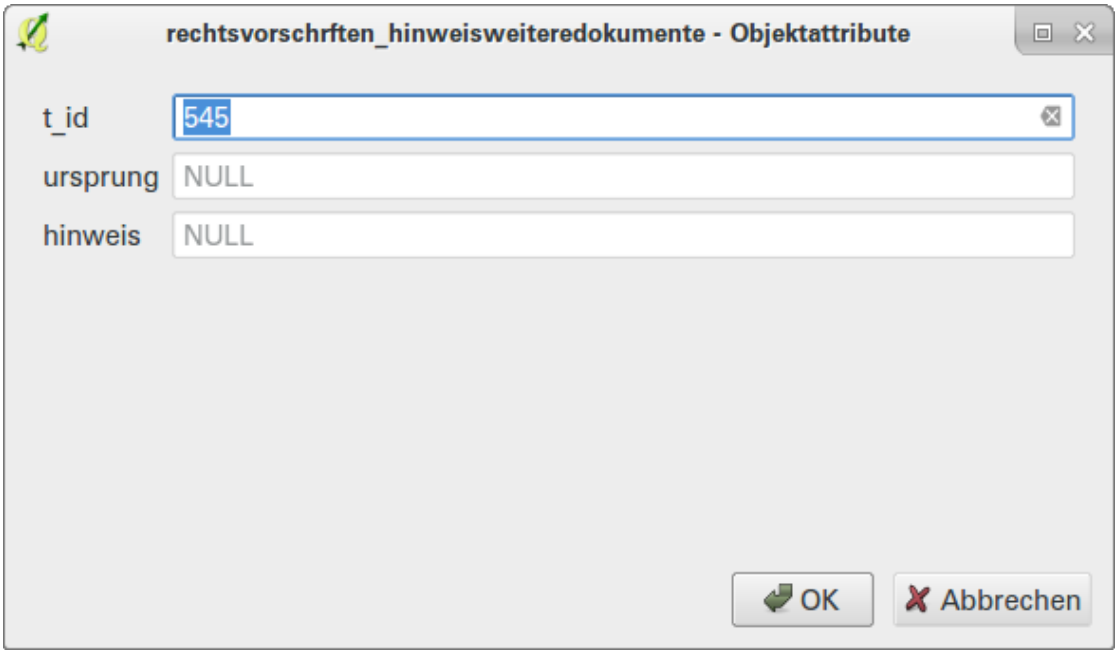


Abbildung 4.17: Erfassung der Beziehung

Noch immer muss man einen nichtssagenden Wert einfüllen. Um das Erfassen jetzt elegant und einfacher zu machen, muss man das passende Bearbeitungselement auswählen:

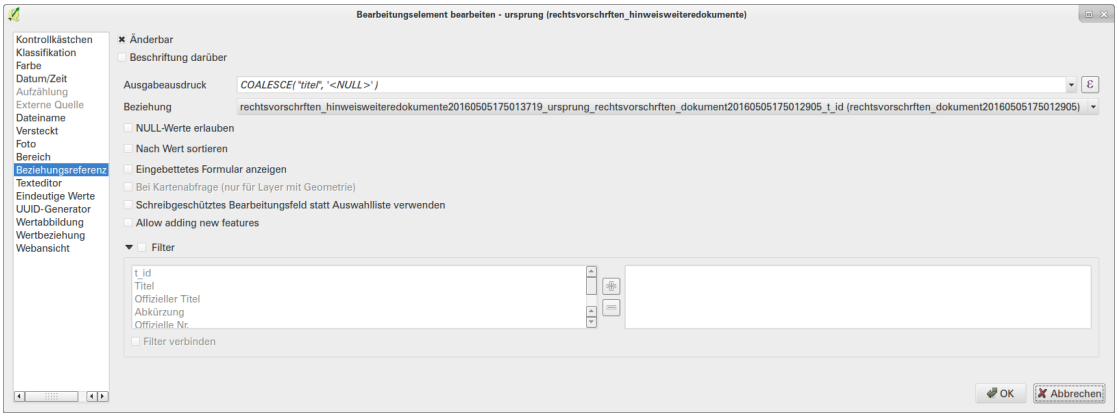


Abbildung 4.18: Bearbeitungselement «Beziehungsreferenz»

Sämtliche Optionen belassen wir in den Standardeinstellungen.

ACHTUNG: Nachfolgend sind einige der Screenshots mit *QGIS 2.14* gemacht. Diese sind speziell mit (*QGIS 2.14*) gekennzeichnet. Der einzige Unterschied ist, der SQL-Ausdruck im Primärschlüssel-Attribut «t_id», der in *QGIS 2.14* noch nicht ausgewertet wird.

Im Eingabeformular erscheinen jetzt Comboboxen und ein zusätzliche Icon rechts davon:

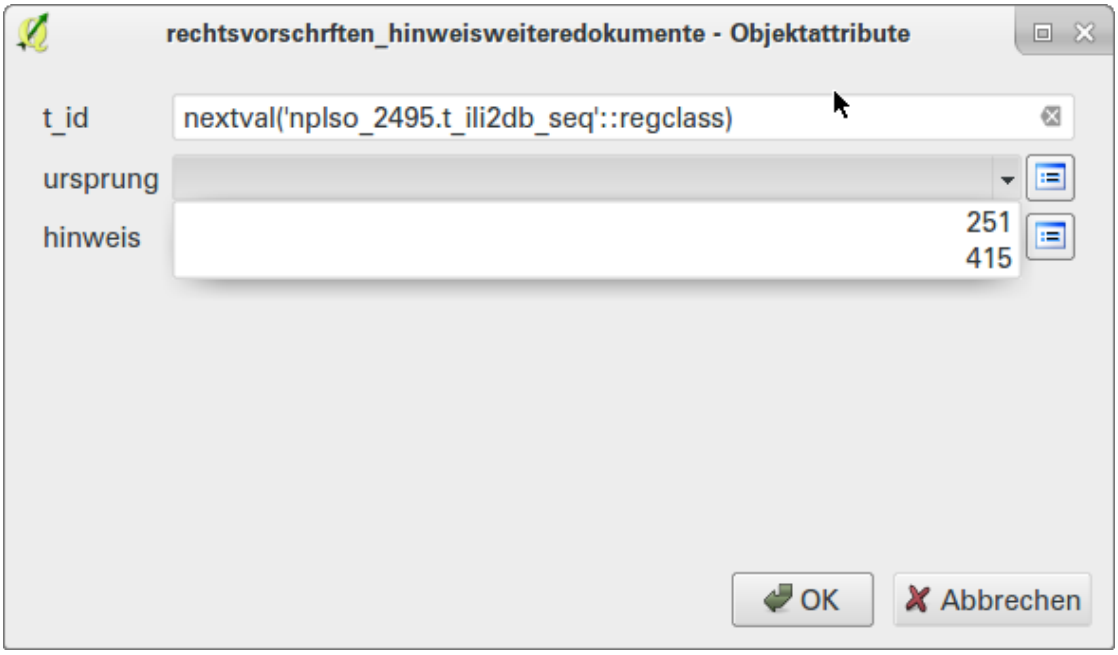


Abbildung 4.19: Eingageformular mit Beziehungsreferenz-Widget (QGIS 2.14)

Richtig aussagekräftig sind die Werte der Combobox jedoch auch nicht. Das kann man schnell ändern, in dem man im Layer «rechtsvorschrften_dokument» in der Attributtabelle in der Formularansicht unter **Ausdruck > Spaltenvorsicht** das gewünschte Attribut wählt, z. B. «title». Unter **Ausdruck > Ausdruck** kann auch eine beliebiger QGIS-Ausdruck[6] stehen, z. B. Kombination aus «title» und «offiziellenr» etc.

[6] http://docs.qgis.org/2.8/de/docs/user_manual/working_with_vector/expression.html

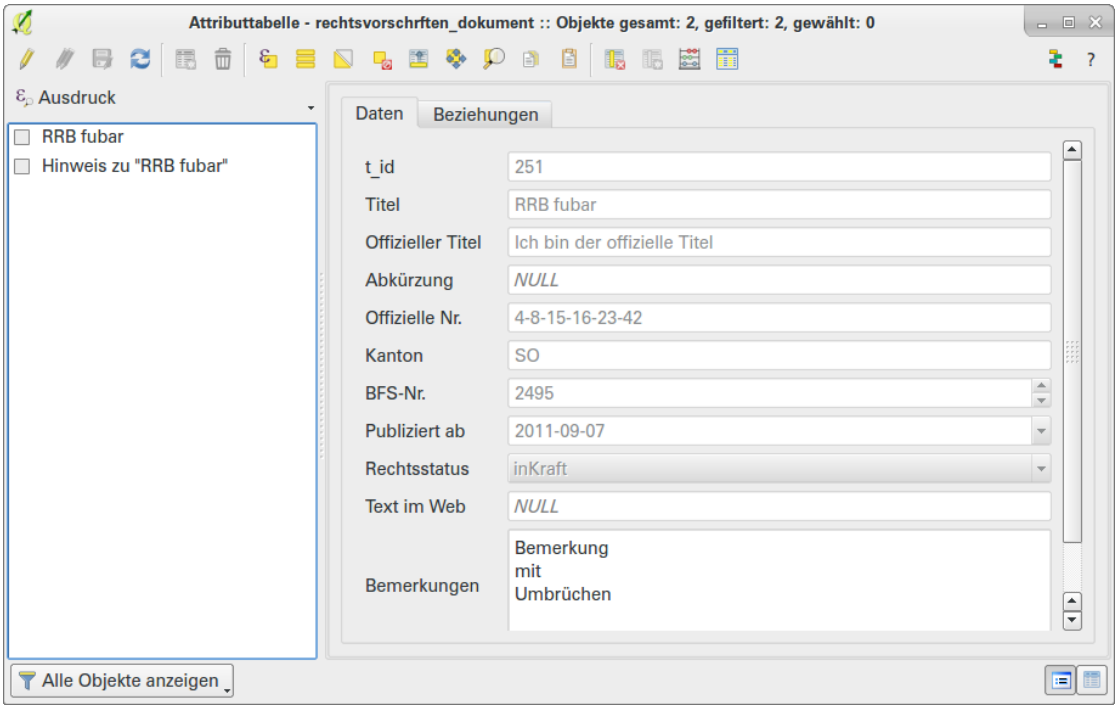


Abbildung 4.20: Anzeige in Formularansicht verändern

Die Combobox im Eingabeformular des Layers «rechtsvorschrften_hinweisweiteredokumente» übernimmt diese Wahl (hier «titel») und das Erstellen der Beziehung fällt nun viel leichter:

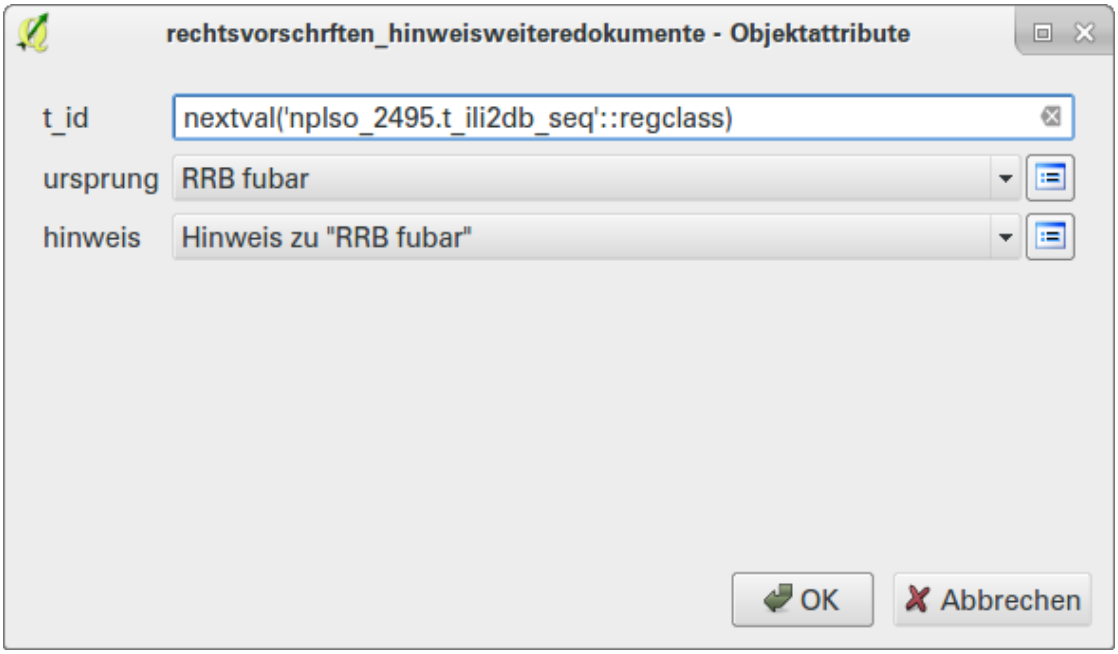


Abbildung 4.21: Eingabeformular mit Beziehungsreferenz-Widget und «sprechender» Auswahl (QGIS 2.14)

Mit dem Icon neben der Combobox lässt sich Eingabeformular des «Ursprungs»- resp. «Hinweis»-Dokumentes öffnen. Mit der Option **Eingebettetes Formular anzeigen** des Bearbeitungselementes «Beziehungsreferenz» lassen sich diese Formulare direkt im Eingabeformular des Layers «rechtsvorschrften_hinweisweiteredokumente» einbetten.

Die Überprüfung in *pgAdmin3* zeigt, dass die Daten auch in der Datenbank gespeichert worden sind:

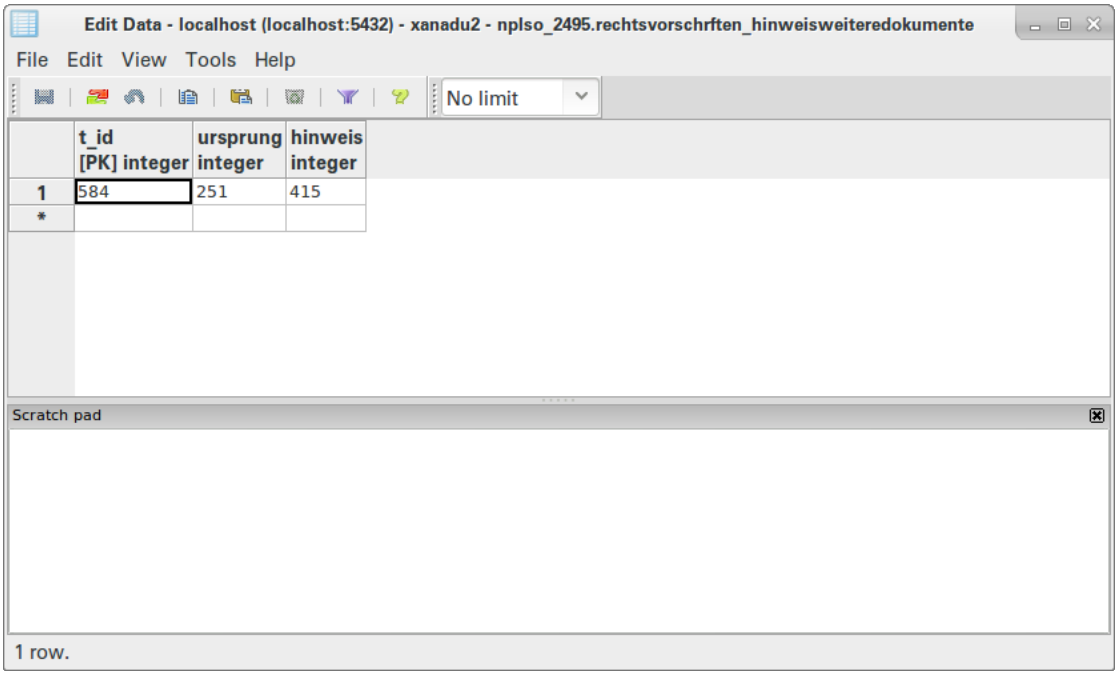


Abbildung 4.22: Beziehung in PostgreSQL gespeichert

4.5 Fachschale

Von der QGIS-Fachschale für die Nutzungsplanung sind wir noch ein gutes Stück entfernt. D.h. wir müssen für alle benötigten Klassen aus dem INTERLIS-Datenmodell die Eingabeformulare individualisieren und die Beziehungen herstellen. Das bedeutet einen gewissen Initialaufwand, der sich aber lohnt, wenn man anschliessend die Daten effizient und qualitativ hochwertig erfassen kann.

Denk- und machbar wäre sicher auch ein Automatismus in *QGIS*, der aus den Informationen aus der Datenbank und eventuell aus den Metadaten-Tabellen («t_ili2db_*») das QGIS-Projekt zusammenstellen kann.

4.6 Erfassung der Grundnutzung

Um auch etwas zum Exportieren zu haben, können wir z. B. für die Grundnutzung die Formulare individualisieren und die Beziehungen herstellen und anschliessend Daten

erfassen.

Betroffen sind folgende INTERLIS-Klassen:

- Typ_Grundnutzung EXTENDS Typ
- Grundnutzung
- Grundnutzung_Pos EXTENDS Pos

Die dazugehörigen Assoziationen:

- Typ_Grundnutzung_Dokument
- Typ_Grundnutzung_Grundnutzung
- Grundnutzung_Grundnutzung_Pos
- Dokument_Grundnutzung

Ein mögliches Vorgehen kurz in Stichworten notiert:

Layer «Typ_Grundnutzung» Das Eingabeformular ist zu individualisieren und einige Daten sind z erfassen, z. B. «W2» und «W2H».

Relation Typ_Grundnutzung_Dokument Diese Beziehung ist zu erstellen. Es ist eine n:m-Beziehung. Bei «cross-refeference tables» ist der «cross-reference table» das «Child». Anschliessend sind die notwendigen Daten zu erfassen.

Layer Grundnutzung Das Eingabeformular ist zu individualisieren (Drag and Drop). Dazu gehören auch zwei Beziehungen (1:m), die erstellt werden müssen. Danach können die Grundnutzungen digitalisiert/erfasst werden. Styling: siehe Kapitel 4.7.1.

4.7 Tipps und Tricks

4.7.1 Styling mit QGIS-Expressions

Normalerweise möchte man die Flächen der Grundnutzung nach ihrer Nutzungszone einfärben. Dieses Attribut ist aber nicht im Layer «nutzungsplanung_grundnutzung», sondern im Layer «nutzungsplanung_typ_grundnutzung». Zwischen beiden Layern existiert eine Beziehung. Dank QGIS-Expressions (deutsch: «Ausdrücke») kann man trotzdem die Polygone wie gewünscht einfärben. Man muss lediglich z. B. im regelbasierten Renderer für die Regeln den passenden Ausdruck verwenden:

```
attribute(get_feature('nutzungsplanung_typ_grundnutzung',
't_id', typ),'abkuerzung')
```

Dieser Ausdruck holt aus dem angegebenen Layer die benötigten Informatione (das Attribut «abkuerzung»). Diese können wir nun in der Regel verwenden:

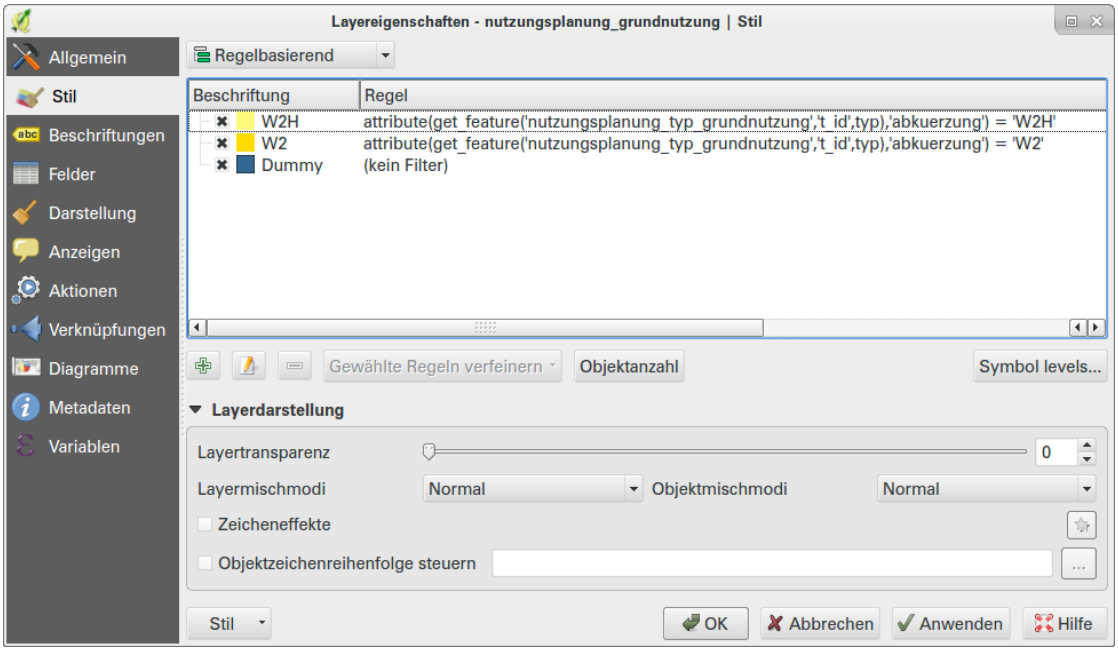


Abbildung 4.23: Styling mit QGIS-Expressions

Wie bereits in Kapitel 2.5 erwähnt, kann das aber zu einem markanten Performance-Downgrade führen.

4.7.2 Multiedit

Ein neues geniales Feature ist das gleichzeitige Editieren von Attributen. Dazu müssen die zu editierenden Objekte selektiert werden. In unserem Beispiel einige der Polygone der Grundnutzung. Anschliessend schaltet man in der Formularansicht in den Editiermodus und zusätzlich in den MultieditModus. Auf der rechten Seite erscheint ein kleines Symbol für jedes Attribut:

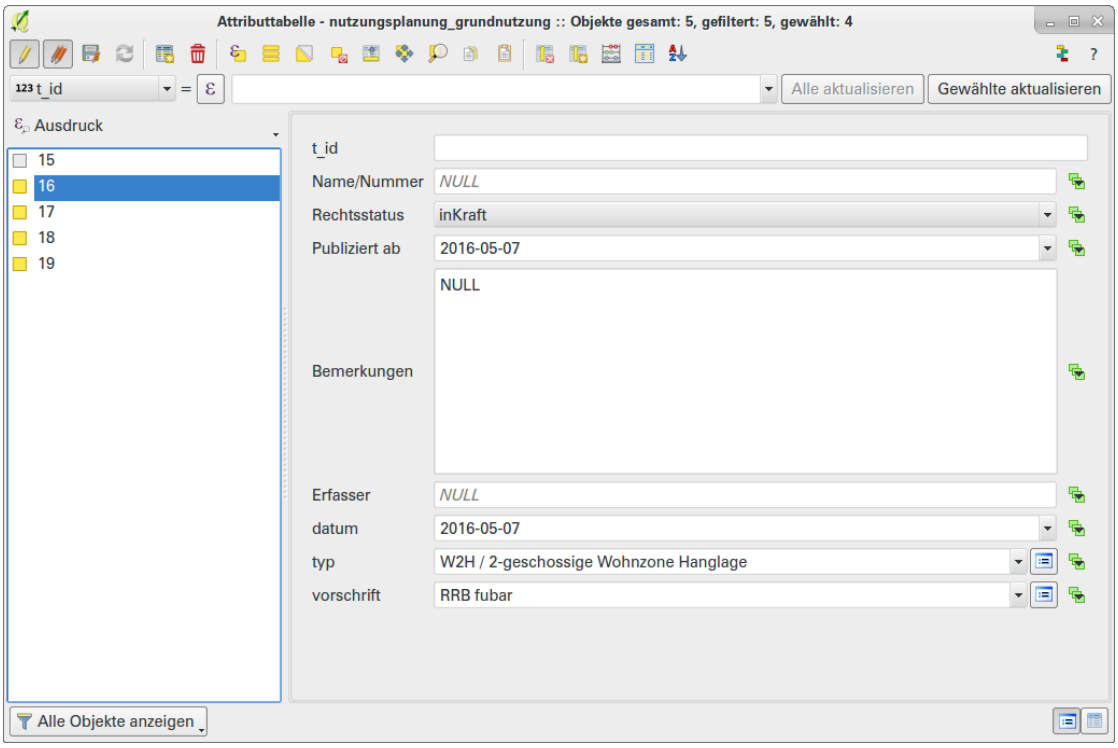


Abbildung 4.24: Multiedit-Modus

Sind sämtliche Attribut-Werte der selektierten Objekte gleich, ist das Symbol grün. Haben die Attribute unterschiedliche Werte, weist das Symbol eine gelbe Farbe auf. Wenn man für sämtliche selektierten Objekte die Nutzungart ändern will, muss man bloss in der Combobox die gewünschte Art auswählen. Das Symbol ändert die Farbe

auf rot und im Formular erscheint aufdringlich ein Meldung, dass es nicht gespeicherte Veränderungen gibt:

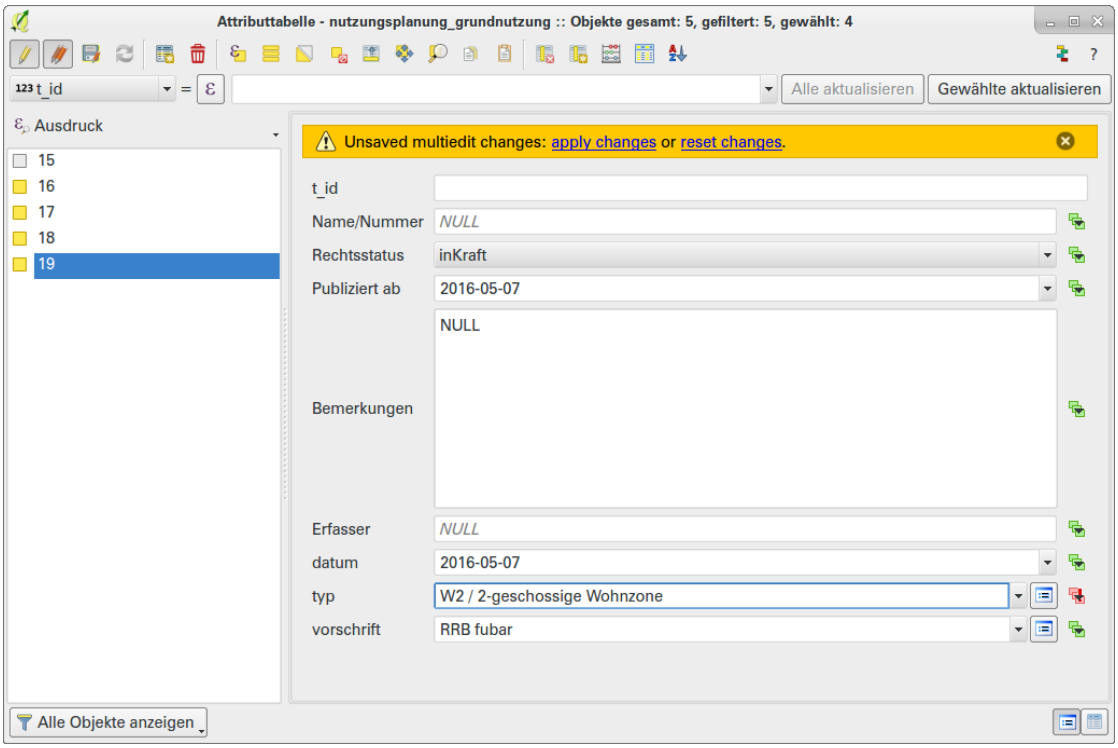


Abbildung 4.25: Veränderte Werte im Multiedit-Modus

Mit «apply changes» können diese Veränderungen angewendet werden. Das Symbol wechselt auf grün.

5 Datenexport

5.1 Ziele

- Daten mit *ili2pg* exportieren
- XML-Datei formatieren

5.2 Daten exportieren

Nachdem wir ein paar Grundnutzungszonen digitalisiert haben, wollen wir unser Werk auch exportieren können. Wie schon in Kapitel 2.6 gezeigt ist das «dead simple»:

```
java -jar ili2pg.jar --dbhost localhost --dbport 5432 --
   dbdatabase xanadu2 --dbusr stefan --dbpwd ziegler12 --models
    SO_Nutzungsplanung_20160427 --dbschema nplso_2495 --export
   daten/nplso/nplso_2495.xtf
```

Bei einem erfolgreichen Export, erscheint auf der Konsole «Info ...export done». Ebenfalls wie beim Import, wird beim Export eine Statistik der exportierten Elemente aufgelistet.

Die erzeugte INTERLIS/XTF-Datei kann man zu Ungunsten der Dateigrösse mit xmllint[1] lesbarer gestalten:

```
xmllint --format daten/nplso/nplso_2495.xtf -o daten/nplso/
   nplso_2495.xtf
```

[1] sudo apt-get install libxml2-utils

Das Ergebnis sieht dann so aus:

```
    </SO_Nutzungsplanung_20160427.Rechtsvorschriften.Dokument>
    <SO_Nutzungsplanung_20160427.Rechtsvorschriften.HinweisWeitereDokumente TID="584">
      <Ursprung REF="251"/>
      <Hinweis REF="415"/>
    </SO_Nutzungsplanung_20160427.Rechtsvorschriften.HinweisWeitereDokumente>
  </SO_Nutzungsplanung_20160427.Rechtsvorschriften>
  <SO_Nutzungsplanung_20160427.Nutzungsplanung BID="SO_Nutzungsplanung_20160427.Nutzungsplanung">
    <SO_Nutzungsplanung_20160427.Nutzungsplanung.Typ_Grundnutzung TID="627">
      <Bezeichnung>2-geschossige Wohnzone</Bezeichnung>
      <Abkuerzung>W2</Abkuerzung>
      <Verbindlichkeit>Nutzungsplanfestlegung</Verbindlichkeit>
      <Geschosszahl>2</Geschosszahl>
      <Typ_Kt>N111_Wohnzone_2_G</Typ_Kt>
      <Code_kommunal>1112</Code_kommunal>
    </SO_Nutzungsplanung_20160427.Nutzungsplanung.Typ_Grundnutzung>
    <SO_Nutzungsplanung_20160427.Nutzungsplanung.Typ_Grundnutzung TID="649">
      <Bezeichnung>2-geschossige Wohnzone Hanglage</Bezeichnung>
      <Abkuerzung>W2H</Abkuerzung>
      <Verbindlichkeit>Nutzungsplanfestlegung</Verbindlichkeit>
      <Geschosszahl>2</Geschosszahl>
      <Typ_Kt>N111_Wohnzone_2_G</Typ_Kt>
      <Code_kommunal>1111</Code_kommunal>
    </SO_Nutzungsplanung_20160427.Nutzungsplanung.Typ_Grundnutzung>
    <SO_Nutzungsplanung_20160427.Nutzungsplanung.Typ_Grundnutzung_Dokument TID="729">
      <Typ REF="627"/>
      <Vorschrift REF="251"/>
    </SO_Nutzungsplanung_20160427.Nutzungsplanung.Typ_Grundnutzung_Dokument>
    <SO_Nutzungsplanung_20160427.Nutzungsplanung.Typ_Grundnutzung_Dokument TID="730">
      <Typ REF="649"/>
      <Vorschrift REF="251"/>
    </SO_Nutzungsplanung_20160427.Nutzungsplanung.Typ_Grundnutzung_Dokument>
    <SO_Nutzungsplanung_20160427.Nutzungsplanung.Grundnutzung TID="774">
      <Geometrie>
        <SURFACE>
          <BOUNDARY>
            <POLYLINE>
              <COORD>
                <C1>2641643.509</C1>
                <C2>1247236.127</C2>
              </COORD>
              <ARC>
                <C1>2641648.443</C1>
                <C2>1247254.329</C2>
                <A1>2641645.366</A1>
                <A2>1247245.393</A2>
              </ARC>
              <COORD>
                <C1>2641655.084</C1>
                <C2>1247248.903</C2>
              </COORD>
              <COORD>
```

Abbildung 5.1: Formatierte INTERLIS/XTF-Datei

Möchte man anstelle von INTERLIS/XTF INTERLIS/GML exportieren, muss man bloss die Dateiendung ändern:

```
java -jar ili2pg.jar --dbhost localhost --dbport 5432 --
   dbdatabase xanadu2 --dbusr stefan --dbpwd ziegler12 --models
    SO_Nutzungsplanung_20160427 --dbschema nplso_2495 --export
   daten/nplso/nplso_2495.gml
```

6 Ausblick

6.1 Open Source INTERLIS-Checker

Bevor irgendwo, irgendwem Daten geliefert werden, müssen sie geprüft werden. Für INTERLIS gibt es heute nicht allzu viele Alternativen. Dies dürfte sich in Zukunft ändern. Bis Ende 2016 / Frühjahr 2017 sollte es einen Open Source INTERLIS-Checker geben.

Geschrieben wird er in Java und kann somit auch sehr einfach z. B. als Webdienst eingesetzt werden.

6.2 ili2gpkg

Neben *ili2pg* gibt es auch noch *ili2gpkg*. *Ili2gpkg* erzeugt eine GeoPackage-Datei und importiert in diese die INTERLIS-Daten. Für die Datenerfassung hat das den Vorteil, dass man unabhängig von einer PostgreSQL/PostGIS-Installation wird.

Für den produktionen Einsatz ist die Kombiniation QGIS-ili2gpkg noch nicht zu empfehlen:

- Die Unterstützung von Kreisbogen in QGIS-GeoPackage ist noch ganz neu.
- Momentan können geometrielose Tabellen von ili2gpkg-erzeugten GeoPackage-Dateien noch nicht in QGIS importiert werden[1].
- Die Performance der Attributtabelle beim Scrollen ist mit ili2gpkg-erzeugten GeoPackage-Dateien noch schlecht.

[1] http://osdir.com/ml/qgis-user-gis/2016-04/msg00264.html

7 Links

7.1 ili2pg

- Generische Umsetzung der minimalen Geodatenmodelle in der kantonalen Geodateninfrastruktur. Fachstelle Geoinformation. Dr. Peter Staub. 2016-02-09.
- ili2pg-Workshop. Oliver Jeker. Peter Staub. Stefan Ziegler. 2016-03-10.
- INTERLIS leicht gemacht. Blog-Reihe. Stefan Ziegler.

7.2 QGIS

- QGIS Relations. Matthias Kuhn. 2013-11-10.