Nutzerdokumentation zum Import von NAS-Dateien mit ogr2ogr in das vollständige AAA-PostNAS-Schema

Erstellt von: Dr. Peter Korduan, GDI-Service
letzte Änderung am: 13.04.2017

Kontakt:
GDI-Service Rostock
Dr. Peter Korduan
Joachim-Jungius-Straße 9
18059 Rostock
Tel: 0381/40344444
E-Mail: peter.korduan@gdi-service.de

Änderungen:

Datum	Änderung
13.04.2017	Auslagerung des Abschnittes „Schritt für Schritt Anleitung“ aus der ursprünglichen Dokumentation zum Projekt in dieses Dokument als Nutzerdokumentation
25.04.2017	Kleine Rechtschreibkorrekturen

Inhaltsverzeichnis

1 Zweck des Dokumentes

Dieses Dokument dienst der Beschreibung des Einlesevorgangs von NAS-Dateien in das vollständiges PostNAS-Schema. Voraussetzung für das Einlesen sind eine PostgreSQL-Datenbank mit PostGIS Erweiterung, das Programm ogr2ogr aus dem GDAL Paket und Ruby für die Umbenennungen in den NAS-Dateien. Für die Installation der Datenbankkomponente und OGR wird auf externe Dokumentationen verwiesen, da dies bei den Anwendern je nach Betriebssystem und Infrastruktur sehr unterschiedlich sein kann. Da die Einrichtung von Ruby und der xml-Bibliothek zusätzlich zu dem sonst üblichen Einleseprozess von NAS-Dateien mit ogr2ogr nötig ist, wird diese in diesem Dokument ausführlich beschrieben.

Die Beschreibung zur Vorbereitung des Einleseprozesses gliedert sich in einen Teil über die einmalige Vorbereitung, die darin besteht das Datenbankschema einzurichten und einen Teil, der die Umbenennungen in den NAS-Dateien beschreibt. Letzterer Schritt kann zusammen mit dem Einlesen mit ogr2ogr in einem Einleseskript automatisiert werden.

An Hand dieser Dokumentation soll der Nutzer befähigt werden selbständig NAS-Dateien in eine PostgreSQL-Datenbank einlesen zu können.

Nicht behandelt wird in diesem Dokument wie das vollständige Schema aus dem AAA-Modell abgeleitet wurde und wie es gefiltert werden kann um einzelne Modellklassen und Attribute wegzulassen. Dazu wird auf den Projektbericht unter http://gdi-service.de/xmi2db/doc/Projektbericht.pdf verwiesen.

2 Installation

2.1 PostgreSQL

Siehe PostgreSQL-Dokumentation zur Server Administration:

https://www.postgresql.org/docs/9.4/static/admin.html

Eine Einfache Variante ist die Einrichtung eines Docker-Containers. Dazu ist lediglich docker erforderlich, was unter Linux mit dem Befehl

```
apt-get update && apt-get install docker
```

installiert wird und der Aufruf:

```
  docker run --name pgsql-server \
    -h pgsql-container \
    -v /home/$LOCAL_USER/postgresql/data:/var/lib/postgresql/data \
    -p 5432:5432 \
    -e POSTGRES_USER=$PGSQL_USER \
    -e POSTGRES_PASSWORD=$PGSQL_PASSWORD \
    -e POSTGRES_DB=$PGSQL_DB \
    -e PG_MAJOR=$PGSQL_MAJOR_VERSION \
```

```
    --restart=always \
    -d pkorduan/postgis:$DOCKER_IMAGE_VERSION
```

Sorgen Sie dafür, dass vorher das Verzeichnis /home/$LOCAL_USER/postgresql/data existiert. Es wird empfohlen als lokalen Nutzer auf dem Hostrechner von Docker gisadmin einzurichten, da dieser Nutzername auch im GDAL-Container verwendet wird, siehe auch Abschnitt 2.3.
Im folgenden hat man einen Container mit einem laufenden PostgreSQL Server, der über den Hostnamen pgsql-server und Port 5432 mit den eingegebenen Zugangsdaten angesprochen werden kann.
Ersetzen sie die Variablen, die mit $ am Anfang gekennzeichnet sind durch eigenen Werte. Für $DOCKER_IMAGE_VERSION wählen Sie den aktuellsten der unter https://hub.docker.com/r/pkorduan/postgis/tags/ angegebenen Tags. Das Datenbankverzeichnis liegt auf dem Hostrechner unter /home/$LOCAL_USER/postgresql/data. Passen Sie dort aus Sicherheitsgründen als erstes die Zugangsberechtigungen in der Datei pg_hba.conf an. Die Authentifizierung sollte mindestens mit der Methode md5 erfolgen und nicht trust. Eine Beschränkung auf bestimmte IP-Adressen wird ebenfalls empfohlen um den Zugang nur von bestimmten Rechnern aus zu erlauben. Führen Sie anschließend ein reload der Konfiguration im oder einen Neustart des Containers aus.

2.2 PostGIS

Siehe PostGIS-Dokumentation:

http://postgis.net/docs/manual-2.1/postgis_installation.html

Falls Sie PostgreSQL mit dem Docker-Image pkorduan/postgis installiert haben, ist die PostGIS Erweiterung schon mit im Container enthalten. Die Einrichtung der Erweiterung in der PostNAS-Datenbank wird in Abschnitt 3.2 behandelt.

2.3 GDAL mit ogr2ogr

ogr2ogr ist ein Hilfsprogramm mit dem GIS-Daten von einem zu einem anderen Datenformat konvertiert werden können. Damit ist auch das Einlesen von Daten in eine PostGIS-Datenbank möglich. Für die Verwendung des OGR-Treibers zum Einlesen von NAS-Dateien, die ja auf GML basieren wird die Installation über den GDAL/OGR-Trunk empfohlen,

siehe http://trac.wheregroup.com/PostNAS/wiki/BuildingOgrPostNASDriver

Allgemeine Informationen zum Installieren von OGR finden sie hier:

http://trac.osgeo.org/gdal/wiki/BuildHints

Eine einfache Variante ist wieder die Einrichtung eines Docker-Containers. Speziell für das Einlesen von NAS-Dateien wurde das Image pkorduan/gdal-sshd erstellt. Ein Docker-Container auf der Basis dieses Images läßt sich folgt erstellen:

```
docker run --name gdal \
    -h gdal-container \
    -v /home/$LOCAL_USER/gdal/data:/var/gdal/data \
    --link pgsql-server:pgsql \
    --restart=always \
    -P \
    -d pkorduan/gdal-sshd:$GDAL_IMAGE_VERSION
```

Sorgen Sie dafür, dass vorher das Verzeichnis /home/$LOCAL_USER/gdal/data existiert. Dieses Verzeichnis wird in den Container an die Stelle /var/gdal/data gemountet. In diesem Verzeichnis können dann die NAS-Dateien, die eingelesen werden sollen abgelegt werden. Dieser Container bindet einen vorhandenen Container pgsql-server ein, in dem ein Postgres-Datenbankserver läuft, siehe Abschnitt 2.1. Wenn Sie schon einen anderen Datenbankserver laufen haben, können Sie den Parameter --link weglassen und bei der Datenbankverbindung, die bei ogr2ogr angegeben werden muss, Ihre Zugangsdaten verwenden. Für $GDAL_IMAGE_VERSION wählen Sie den aktuellsten Tag von der Seite https://hub.docker.com/r/pkorduan/gdal-sshd/tags/ aus.
Sie haben nun einen laufenden Container in dem auch schon libxml-ruby installiert ist, siehe Abschnitt 2.4.

Um ogr2ogr mit diesem Image verwenden zu können, braucht der Container nicht unbedingt als Service zu laufen. Mit dem Parameter -d existiert der Container aber permanent, Sie können sich mit ssh auf diesen Container verbinden und Programme darin mit docker exec ausführen, siehe Abschnitt 4.2. Im Container steht auch der Postgres Client psql zur Verfügung.

2.4 Installation libxml-ruby

Um NAS-Dateien in das vollständige PostNAS-Schema einlesen zu können, müssen einige XML-Elemente in den GML-Dateien umbenannt werden. Dazu wurde das Ruby-Programm rename_nas.rb geschrieben, welches unter dem folgenden Link heruntergeladen werden kann:

http://gdi-service.de/xmi2db/converter/rename_nas.rb

Die Ausführung unter Debian 7 erfordert die Installation von libxml-ruby. Dieses Debian-Paket wird wie folgt installiert:

```
apt-get update && apt-get install \
  ruby1.9.3 \
  ruby1.9.3-dev \
  libxml-ruby
gem install libxml-ruby
```

Bei Verwendung von rvm die Umgebung setzen.

```
source /etc/profile.d/rvm.sh
```

Falls die Installation von libxml-ruby nicht funktioniert liegt das eventuell an einer veralteten Version. Libxml-ruby sollte mindestens die Version 2.9.0 haben. Das kann abgefragt werden durch Eingabe von:

```
puts XML::VERSION
```

Am Anfang von rename_nas.rb und Aufruf mit:

```
ruby rename_nas.rb
```

Unter Debian 8 heißt die Bibliothek ruby-libxml.
Wer das Docker-Image pkorduan/gdal-sshd ab Version 2.2.1 verwendet hat libxml-ruby bereits zusammen mit ogr2ogr aus dem OGR-Trunk vorinstalliert zur Verfügung.

3 Einlesen vorbereiten

Wenn eine PostgreSQL-Datenbank, ogr2ogr sowie Ruby installiert sind kann mit der Vorbereitung des Einlesevorganges begonnen werden. Als einmaliger Vorgang wird das Anlegen der Datenbank mit PostGIS-Erweiterung und das Erzeugen des vollständigen PostNAS-Schemas betrachtet, siehe Abschnitte 3.1 und 3.2. Ein Vorgang, der für jede einzulesende Datei wiederholt werden muss ist das Umbenennen von Tags in den NAS-Dateien, siehe Abschnitt 3.3.

3.1 Einrichtung der Datenbank mit PostGIS-Extension

Die Datenbank wird mit dem SQL-Befehl

```
CREATE DATABASE alkis
```

erzeugt. Als Datenbank-Client kann man z.B. pgadmin3[1] mit graphischer Oberfläche oder den Konsole-Client psql[2], welcher im GDAL-Container läuft, verwenden. Ersetzen Sie den Namen alkis durch einen anderen, wenn Ihre Datenbank anders heißen soll. In den folgenden Beispielen wird immer der name alkis für die Datenbank verwendet.

Zum Hinzufügen der PostGIS-Extension folgenden SQL-Befehl ausführen:

```
CREATE EXTENSION postgis;
```

1 pgadmin3: https://www.postgresql.org/ftp/pgadmin3/release/
2 psql: https://www.postgresql.org/docs/9.4/static/app-psql.html

Anschließend findet sich im public Schema eine Tabelle spatial_ref_sys, siehe Abbildung 1.

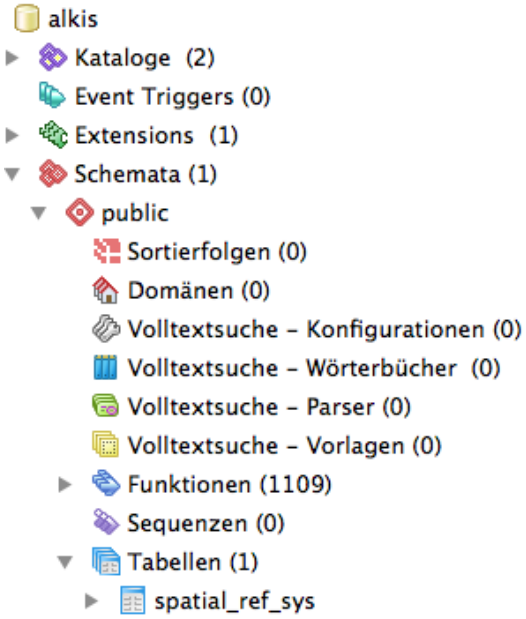

Abbildung 1: Tabelle spatial_ref_sys im Schema public nach Anlegen der PostGIS Erweiterung

3.2 Erstellung des Schemas "aaa_ogr"

Das vollständige Schema kann unter

http://gdi-service.de/xmi2db/sql/schema_aaa_ogr_25833.sql

heruntergeladen werden. Ersetzen Sie 25833 durch 25832 wenn Sie die Tabellen mit dem Koordinatenreferenzsystem ETRS 89 / Zone 32 haben möchten.

Der Befehl zum Ausführen der SQL-Datei im Console-Client psql lautet:

```
psql -h $PGSQL_HOST -U $PGSQL_USER -f schema_aaa_ogr_25833.sql alkis
```

Ersetzen Sie $PGSQL_HOST und $PGSQL_USER durch Ihre Zugangsdaten. Bei Verwendung der Docker-Container ist $PGSQL_HOST pgsql. Wird der Parameter -h weggelassen wird localhost angenommen. Siehe psql --help für mehr Informationen. Alternativ kann der SQL-Text auch in den SQL-Editor von pgadmin3 kopiert und dort ausgeführt werden.

Das erzeugte Schema heißt standardmäßig aaa_ogr. Möchte man dass das Schema anders heißen soll, muss man vorher in der Datei schema_aaa_ogr_25833.sql alle Textstellen wo aaa_ogr vorkommen durch seinen Namen ersetzen. Im Folgenden wird davon ausgegangen, dass das Schema aaa_ogr heißt. Passen Sie die Befehle entsprechend an, wenn Sie Ihr Schema anders genannt haben.

Die Ausführung der SQL-Datei kann länger als eine Minute dauern, da 490 Tabellen angelegt und die Aufzählungstabellen mit Werten gefüllt werden müssen, siehe Abbildung 2.

Das aktuelle Schema befindet sich im Repository der Anwendung xmi2db. Es bietet sich an das Repository gleich komplett herunterzuladen entweder über git:

```
git clone https://github.com/pkorduan/xmi2db.git
```

oder per Download von github.com über den Link:

https://github.com/pkorduan/xmi2db/archive/master.zip

Darin enthalten sind neben dem Schema und der Filterdatei auch die Umbenennungsliste und das Umbenennungsskript, welches wir für die Vorverarbeitung der NAS-Dateien, siehe Abschnitt 3.3 benötigen.

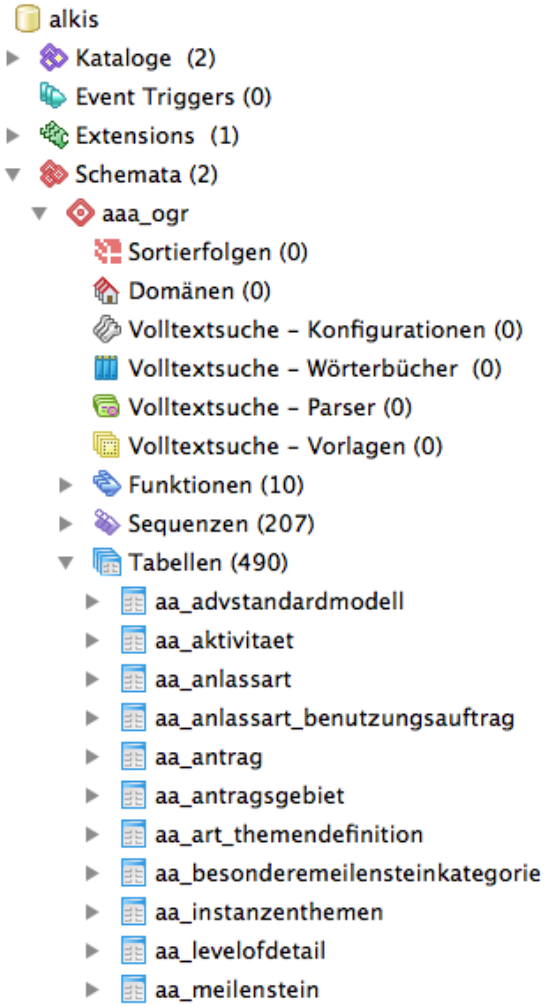

Abbildung 2: Tabellen im flachen OGR-Schema

3.3 NAS-Datei aufbereiten

Zum Umbenennen einer NAS-Datei wird das Umbenennungsskript rename_nas.rb verwendet, welches unter http://gdi-service.de/xmi2db/converter/rename_nas.rb zu finden ist. In dieser Datei ist der Pfad zur Umbenennungsdatei angegeben. Dieser geht standardmäßig davon aus, dass die Umbenennungsdatei umbenenn_conf.json in einem Verzeichnis conf unterhalb des Elternverzeichnisses von rename_nas.rb enthalten ist, so wie in der Verzeichnisstruktur von xmi2db, siehe Abbildung 3.

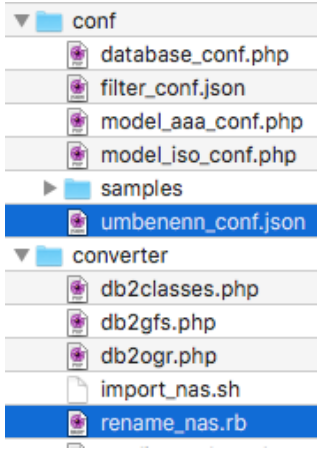

Abbildung 3: Relative Lage von umbenenn.conf zu rename_nas.rb

Jede NAS-Datei, die in das Schema aaa_ogr eingelesen werden soll, muss vorher mit dem Script "rename_nas.rb" aufbereitet werden. Zur Installation von ruby siehe Abschnitt 2.4.

Die Umbenennung von Elementen in einer NAS-Datei "eingabedatei.xml" wird wie folgt aufgerufen:

```
ruby rename_nas.rb eingabedatei.xml [ausgabedatei.xml]
```

Der Name der Ausgabedatei kann optional angegeben werden. Wird dieser Parameter weggelassen, wird die Ausgabedatei automatisch eingabedatei_renamed.xml genannt und im gleichen Verzeichnis wie die Eingabedatei abgelegt. Die Dateinamen können auch mit Pfadangaben versehen werden.
Die erzeugte Ausgabedatei kann dann wie in den Abschnitten 4.1 und 4.2 beschrieben in die Datenbank eingelesen werden.

4 Einlesen

4.1 Eine einzelne NAS-Datei einlesen

Eine einzelne aufbereitete NAS-Datei "renamed_nas.xml" wird wie folgt mit ogr2ogr in das Schema "aaa_ogr" eingelesen.

```
ogr2ogr -f "PostgreSQL" --config PG_USE_COPY NO -nlt CONVERT_TO_LINEAR
-append PG:"dbname=alkis active_schema=aaa_ogr user=$PGSQL_USER
host=$PGSQL_HOST port=5432" -a_srs EPSG:25833 renamed_nas.xml
```

Ersetzen Sie hier wieder die Werte für die Parameter je nach Bedarf. Beachten Sie, dass die Datei renamed_nas.xml in dem Verzeichnis liegen muss in dem Sie den Befehl ausführen. Sie können für die Datei auch einen Pfad angeben.

Siehe http://gdal.org/drv_nas.html für mehr Informationen zur Benutzung von ogr2ogr.

4.2 Einlesen mit Docker-Container

Der gleiche Befehl mit Docker-Container lautet:

```
docker exec gdal ogr2ogr -f "PostgreSQL" --config PG_USE_COPY NO -nlt
CONVERT_TO_LINEAR -append PG:"dbname=alkis active_schema=aaa_ogr
user=$PGSQL_USER
```

Sie können sich auch erst im Container mit einer Konsole anmelden

```
docker exec -it gdal /bin/bash
```

und führen den Befehl dann in dem Container wie oben geschrieben aus.

4.3 Automatisierung des Einlesens von Massendaten

NAS-Dateien, die im nutzer- oder stichtagsbezogenem Abgabeverfahren (NBA) von AAA-Softwaresystemen erzeugt werden, liegen in der Regel in Form von gepackten und komprimierten Archiven vor, z.B. NBA_Grundausstattung_2015-02-11.zip.

Unter Linux lassen sich solche Archive wie folgt entpacken:

```
unzip NBA_Grundausstattung_2015-02-11.zip
```

Es entstehen viele Dateien z.B.

```
NBA_Grundausstattung_001_081_2015-02-11.xml.gz
NBA_Grundausstattung_002_081_2015-02-11.xml.gz
...
NBA_Grundausstattung_081_081_2015-02-11.xml.gz
```

Diese Dateien wiederum lassen sich wie folgt entpacken und in einer Schleife verarbeiten.

```
gunzip *.xml.gz
for NAS_FILE in *.xml
```

```
do
  ruby rename_nas.rb $NAS_FILE renamed_nas.xml
   ogr2ogr -f "PostgreSQL" --config PG_USE_COPY NO -nlt CONVERT_TO_LINEAR
-append PG:"dbname=alkis active_schema=aaa_ogr user=$PGSQL_USER
host=$PGSQL_HOST port=5432" -a_srs EPSG:25833 renamed_nas.xml
done
```

In der Schleife der Abarbeitung ist jedoch noch zu berücksichtigen, dass die erste Datei Metadaten enthält und ignoriert werden kann und Fehler abgefangen werden müssen.

Ein Vorschlag für ein Bash-Skript für Linux, welches die Metadaten und Fehlerbehandlung berücksichtigt in Log-Dateien protokolliert und abgearbeitete Dateien in einen Archivordner schreibt, findet sich in der folgenden Datei http://gdi-service.de/xmi2db/converter/import_nas.sh.

Die Datei muss im Modus "Ausführbar" sein

```
chmod a+x import_nas.sh
```

Passen Sie vor dem Ausführen der Datei mit

```
./import_nas.sh
```

die folgenden Parameter an.

```
DATA_PATH="/pfad/zu/den/nas/dateien"
OGR_PATH="/pfad/zu/ogr2ogr/bin/verzeichnis"
ERROR_LOG=/pfad/fuer/logfiles/mylogfile.log
```

5 Post Processing

Alle weiteren Schritte, die nach dem Einlesen notwendig sind um die Datenbank für weitere Anwendungen gebrauchsfähig zu machen sind nicht Gegenstand dieser Dokumentation.

Siehe dazu z.B. die Dokumentationen des WebGIS kvwmap unter kvwmap.de

http://kvwmap.de/index.php/Installation_von_PostNAS_und_Zugriff_mit_kvwmap

oder das Wiki von PostNAS: http://trac.wheregroup.com/PostNAS/wiki/

6 Nutzerspezifische Schemata

6.1 Gefiltertes Schema erzeugen

Wer nicht alle Elemente des Schemas möchte, kann diese im Schema selbst löschen oder ein eigenes gefiltertes Schema mit xmi2db erzeugen. Dazu wird zunächst xmi2db auf einen Web-Server installiert:

1. Clone das Projekt in das eigene Verzeichnis.

```
git clone https://github.com/pkorduan/xmi2db.git
```

2. Erzeuge und editiere die Datei database_config.php

```
cp conf/database_conf_sample.php conf/database_conf.php
```

3. Passe Datenbankzugang an: PG_HOST, PG_USER, PG_PASSWORD, PG_DBNAME
4. Erzeuge eine Datenbank, die $PG_USER gehört und installiere die Erweiterung PostGIS
5. Lege die zu importierende XMI-Datei im Unterordner xmis ab.
6. Öffne den Link um auf die Konvertierungsoberfläche zu kommen. http://yourserver.de/xmi2db/

Zur Anpassung des Filters kann die Datei conf/filter_conf.json angepasst werden oder eine andere angepasste Filterdatei im Parameter FILTER_FILE hinterlegt werden.

Derzeit werden in filter_conf.json nur die Attribute objektkoordinaten der Klassen *AX_Flurstueck*, *AX_HistorischesFlurstueckALB*, *AX_HistorischesFlurstueckOhneRaumbezug* und *AX_HistorischesFlurstueck* rausgefiltert, weil ogr2ogr nicht mit mehr als einer Geometriespalte umgehen

kann.

Sollten weitere Objekte gefiltert werden und lassen sich diese pro Bundesland bündeln, werden diese in den Dateien conf/samples/filter_[mv | rlp | sl | … | etc].conf abgelegt.

Es können immer nur ganze Objekte oder Attribute von Objekten gefiltert werden. Weitere Informationen dazu in der Projektdokumentation Abschnitt Filter.

6.2 Umbenennungsliste erzeugen

Die aktuelle Umbenennungsliste des vollständigen Schemas ist erhältlich unter dem Link:

https://gdi-service.de/xmi2db/conf/umbenenn_conf.json

Wer ein nutzerspezifisches Schema erstellt hat, sollte auch die Umbenennungsliste anpassen über den Aufruf in xmi2db:

https://gdi-service.de/xmi2db/listings/umbenennungsliste.php

Werden z.B. bestimmte Klassen oder/und Attribute herausgefiltert, müssen diese gegebenenfalls auch nicht in den NAS-Dateien umbenannt werden. Alle Objekte, die nicht im Datenbankschema enthalten sind, werden von OGR auch nicht eingelesen, müssen daher auch nicht vorher umbenannt werden. Aber Vorsicht beim Filtern von Attributen, die mehrfach im Objektbaum vorkommen. Dazu ein Beispiel:

Im Objekt *AX_Flurstueck* gibt es u.a. die Pfade
AX_Flurstueck_gemarkung_AX_Gemarkung_Schluessel_land
AX_Flurstueck_gemeindezugehoerigkeit_AX_Gemeindekennzeichen_land
AX_Flurstueck_zustaendigeStelle_AX_Dienststelle_Schluessel_land

Das erste wird umbenannt nach *gemarkung_land*, das zweite nach *gemeindezugehoerigkeit_land* und das dritte nach *zustaendigestelle_land*.

Werden jetzt z.B. die Attribute *gemeindezugehoerigkeit* und *zustaendigeStelle* herausgefiltert, wird das Attribut *land* in *AX_Flurstueck_gemarkung_AX_Gemarkung_Schluessel_land* nicht mehr umbenannt, da es land ja jetzt nur noch einmal gibt.
Wird nun die Umbenennungsliste neu geschrieben, werden alle 3 vorkommen von land nicht mehr umbenannt. Wenn ogr2ogr dann die NAS-Daten einließt kann das Attribut land, was zu gemarkung gehört nun aber vom gleichnamigen Attribut, welches zu gemeindezugehoerigkeit oder zustaendigestelle gehört überschrieben werden. Daher sollte in diesen Fällen entweder die Umbenennungen der beiden gefilterten Attribute beibehalten werden oder einfach die Spalten im Schema manuell gelöscht werden.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

GeoInfoDok Adv, Dokumentation zur Modellierung der Geoinformationen des amtlichen Vermessungswesens (GeoInfoDok) Hauptdokument Version 6.0.1 Stand: 01.07.2009