kanton glarus

Bau und Umwelt
Raumentwicklung und Geoinformation
Kirchstrasse 2
8750 Glarus

AKTENNOTIZ
vom 2020-08-25
in Sachen **Behälter und Objekte – Implementierung von INTERLIS-Modellen**
Verfasser **Peter Staub, Fachstellenleiter**
Ablage Aktenplan Bau und Umwelt – 479_Uebrige-Aufgaben

Behälter und Objekte – Implementierung von INTERLIS-Modellen

Identifikation von Modellelementen

Die Identifikation von Modellelementen dient verschiedenen Zwecken. In erster Näherung ist die momentane Integrität des Datenbankbestands sicherzustellen. Dazu dienen die so genannten «Primärschlüssel», «Primary Keys» (PK) in Datenbanktabellen. Darüber hinaus dienen so genannte stabile Objekt-Identifikatoren (OID) der langfristigen Identifikation von Objekten auch ausserhalb der originären Datenbanktabelle, etwa in einer Datei während eines Datentransfers. PK werden immer wieder neu erzeugt, etwa bei einem Datenimport. OID müssen hingegen über die gesamte Lebensdauer eines Objektes unverändert bleiben. Sie sind zwingende Voraussetzung für inkrementelle Datenaktualisierungen.

Im Zusammenhang mit INTERLIS-Datenmodellen interessieren in erster Linie die Identifikatoren von Klassenobjekten und Themenobjekten, so genannten Behältern. Behälter repräsentieren ein Modellthema (TOPIC), wobei pro Modellthema je nach Anwendungsfall auch mehrere Behälter definiert werden können, beispielsweise, wenn es in einem Gebiet mehrere Operate gibt.

In einem INTERLIS-Datenmodell gibt es keinen Zwang, eigentliche OID zu definieren. Der Formalismus sorgt quasi automatisch dafür, dass bei jeder Serialisierung von Datendateien «Transfer-Identifikatoren» (TID) für Klassenobjekte und «Behälter-Identifikatoren» (BID) für Behälterobjekte erzeugt werden. Auf die gleiche Weise wie die PK in Datenbanktabellen stellen die TID und die BID nur die momentane Integrität des konkret vorliegenden Datensatzes sicher. Innerhalb einer Datendatei haben BID und TID die Qualität von PK.

Um stabile OID zu erhalten, können in INTERLIS-Datenmodellen OID-Deklarationen pro Thema oder pro Klasse vorgenommen werden, oder es können auch OID-Klassenattribute definiert werden. INTERLIS stellt verschiedene OID-Mechanismen bereit, aus praktischen Gründen wird hier aber nur die Variante mittels *Universally Unique Identifiers* (UUID) – spezifisch: Version 4 – weiterverfolgt. Ein UUID Version 4 hat die Form

```
27b6873c-0b8a-414b-ac1e-0b5c1a0d3961
```

und kann auf vielfältigste Weise erzeugt werden, insbesondere (siehe nächster Abschnitt) mittels Datenbankfunktion `uuid_generate_v4()`.

OID-Varianten in INTERLIS-Datenmodellen

```
MODEL MyModel =
  TOPIC MyTopic = OID AS INTERLIS.UUIDOID;
    CLASS MyClass = OID AS INTERLIS.UUIDOID;
      StableIdent : MANDATORY INTERLIS.UUIDOID;
    END MyClass;
  END MyTopic;
END MyModel.
```

Variante 1: OID-Deklaration für das ganze Thema

Variante 2: OID-Deklaration nur für diese Klasse

Variante 3: OID-Definition als Klassenattribut

Modellkonfiguration mit ili2db (Beispiel PostGIS)

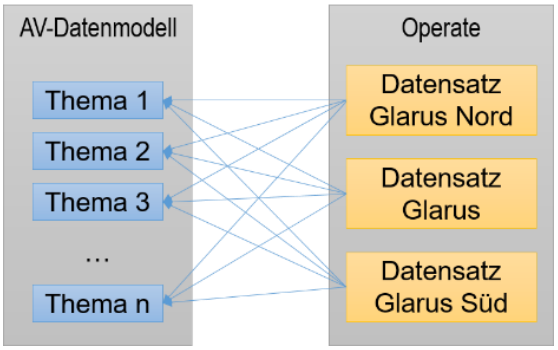
Die generische Schnittstellenbibliothek ili2db unterstützt den Umgang mit OID sowohl für Klassenobjekte als auch für Behälter. Wenn in einem INTERLIS-Datenmodell OID-Deklarationen oder -Definitionen vom Typ `INTERLIS.UUIDOID` vorgenommen wurden, wird das entsprechende Datenbankschema so konfiguriert, dass Tabellenattribute «`t_ili_tid`» vom Typ «`uuid`» und mit der Funktionalität «`DEFAULT uuid_generate_v4()`» erzeugt werden.

Wenn empfangene Datendateien UUID-OID als TID enthalten, kann beim Datenimport mit der Kommandozeilenoption `--importTid` verlangt werden, dass diese in die Datenbank übernommen werden. Beim Datenexport kann umgekehrt mit der Kommandozeilenoption `--exportTid` dafür gesorgt werden, dass die UUID-OID als TID in die Datendatei geschrieben werden anstelle von generischen, nicht stabilen TID.

Datensätze, Behälter und OID

In einem Datenbankschema können verschiedene Behälter, aber auch verschiedene «Datensätze» verwaltet werden.

Beispiel amtliche Vermessung (AV): Das Datenmodell besteht aus mehreren Themen. Im Kanton Glarus bildet jede Gemeinde ein Operat, es gibt also drei Datensätze, die jeweils Daten zu allen Themen enthalten.



In der Datenbank sollen die Daten eines Operats (sprich: einer Gemeinde) als Ganzes behandelt werden können. Es ist zu verhindern, dass etwa bei einem Datenimport bereits vorhandene Daten der anderen Gemeinden gelöscht werden.

Als Vorbereitung muss in der Tabelle `t_ili2db_dataset` pro gewünschtem Operat ein Eintrag erfasst werden. Beim Datenimport und beim Datenexport wird dann auf diesen Eintrag Bezug genommen: `--dataset MyDatasetName`.

Pro Datenthema im Datenmodell wird ein Behälter in der Datenbank angelegt. Wenn man nun Datensätze definiert hat, werden auch die Behälter entsprechend stratifiziert. Analog zu Datenexport und Datenimport von Datensätzen können beispielsweise einzelne Behälter exportiert werden. Dazu ist die Identifikation des gewünschten Behälters zu referenzieren.

In der folgenden Abbildung ist die Tabelle der Behälter (Spalte `topic`) für den Datensatz «1632» (Gemeinde Glarus; Referenz in der Spalte `dataset`) für das Datenmodell der AV dargestellt:

t_id [PK] bigint	dataset bigint	topic character varying(200)	t_ili_tid character varying(200)
2	1	DM01AVCH24LV95D.FixpunkteKategorie1	
57	1	DM01AVCH24LV95D.FixpunkteKategorie2	
113	1	DM01AVCH24LV95D.FixpunkteKategorie3	
1771	1	DM01AVCH24LV95D.Bodenbedeckung	
76266	1	DM01AVCH24LV95D.Einzelobjekte	
90054	1	DM01AVCH24LV95D.Hoehen	
90432	1	DM01AVCH24LV95D.Nomenklatur	
92369	1	DM01AVCH24LV95D.Liegenschaften	
230633	1	DM01AVCH24LV95D.Rohrleitungen	
230635	1	DM01AVCH24LV95D.Nummerierungsbereiche	
230732	1	DM01AVCH24LV95D.Gemeindegrenzen	
234935	1	DM01AVCH24LV95D.Bezirksgrenzen	
234936	1	DM01AVCH24LV95D.Kantonsgrenzen	
234938	1	DM01AVCH24LV95D.Landesgrenzen	
234939	1	DM01AVCH24LV95D.Planeinteilungen	
235198	1	DM01AVCH24LV95D.TSEinteilung	
235231	1	DM01AVCH24LV95D.Rutschgebiete	
235232	1	DM01AVCH24LV95D.PLZOrtschaft	
235265	1	DM01AVCH24LV95D.Gebaeudeadressen	
245585	1	DM01AVCH24LV95D.Planrahmen	

Für die beiden anderen Gemeinden entstünden nochmals die gleichen Einträge in der Tabelle mit der jeweiligen Datensatz-Referenz. Im AV-Datenmodell sind keine stabilen OID definiert, weshalb hier die Spalte `t_ili_tid` leer bleibt.

Externe Kataloge, die als «erweiterter Teil» eines Datenmodells betrachtet werden können und typischerweise zusammen mit den Datenmodellen in der Datenmodellablage publiziert sind, werden als eigene Datensätze behandelt.

Nach der Schemakonfiguration gehört der Import von Katalogdaten auch noch zur Modellimplementierung. Für die Katalogdaten wird ein eigener Datensatz erfasst, damit bei einem späteren Datenexport die eigentlichen Geobasisdaten nicht immer zusammen mit den Katalogdaten unnötigerweise exportiert und transferiert werden.

In folgender Abbildung sind die Datenbanktabellen für die Datensätze und für die Behälter aus dem Schema für das Modell «Kataster der belasteten Standorte» dargestellt.

t_ili2db_dataset: Datensätze (1 = externe Katalogdaten, 23 = Geobasisdaten)

t_id [PK] bigint	datasetname character varying(200)
1	codetexte
23	kbs_kanton_glarus

t_ili2db_basket: Behälter gemäss Themendefinition im Datenmodell

t_id [PK] bigint	dataset bigint	topic character varying(200)	t_ili_tid character varying(200)	attachmentkey character varying(200)
2	1	KbS_Basis_V1_4.Codelisten		KbS_Codetexte_V1_4.xml-2
24	23	GL_KbS_V1.Belastete_Standorte	7714be22-ddb9-4f0d-b2cd-43faa80301fe	x

Projektkonfiguration mit dem QGIS Model Baker

Der QGIS Model Baker ermöglicht die Schemakonfiguration gemäss einem vorgegebenen Datenmodell sowie den Import und Export von INTERLIS-Transferdaten aus QGIS heraus «mittels Knopfdruck» auf sehr einfache Weise. Die eigentliche Kernfunktionalität des QGIS Model Bakers ist das Generieren eines Projekts aus einem modelläquivalenten Datenbankschema. Dabei werden die Modelldefinitionen (Attributtypen, Konsistenzbedingungen, Beziehungen) soweit möglich in den Erfassungsformularen aufbereitet.

Bisher ist es nicht möglich, damit gezielt einzelne Datensätze oder Behälter anzusprechen.

Im Kanton Glarus wird der QGIS Model Baker «nur» zum Generieren von Projektdateien verwendet. Die Schemakonfiguration sowie der produktive Datenimport und die Bereitstellung erfolgen im Rahmen von Integrationsprozessen über das Kommandozeilenprogramm ili2pg.

Aus Anwendersicht bestehen im Zusammenhang mit Datensätzen und Behältern folgende Anforderungen an den QGIS Model Baker:

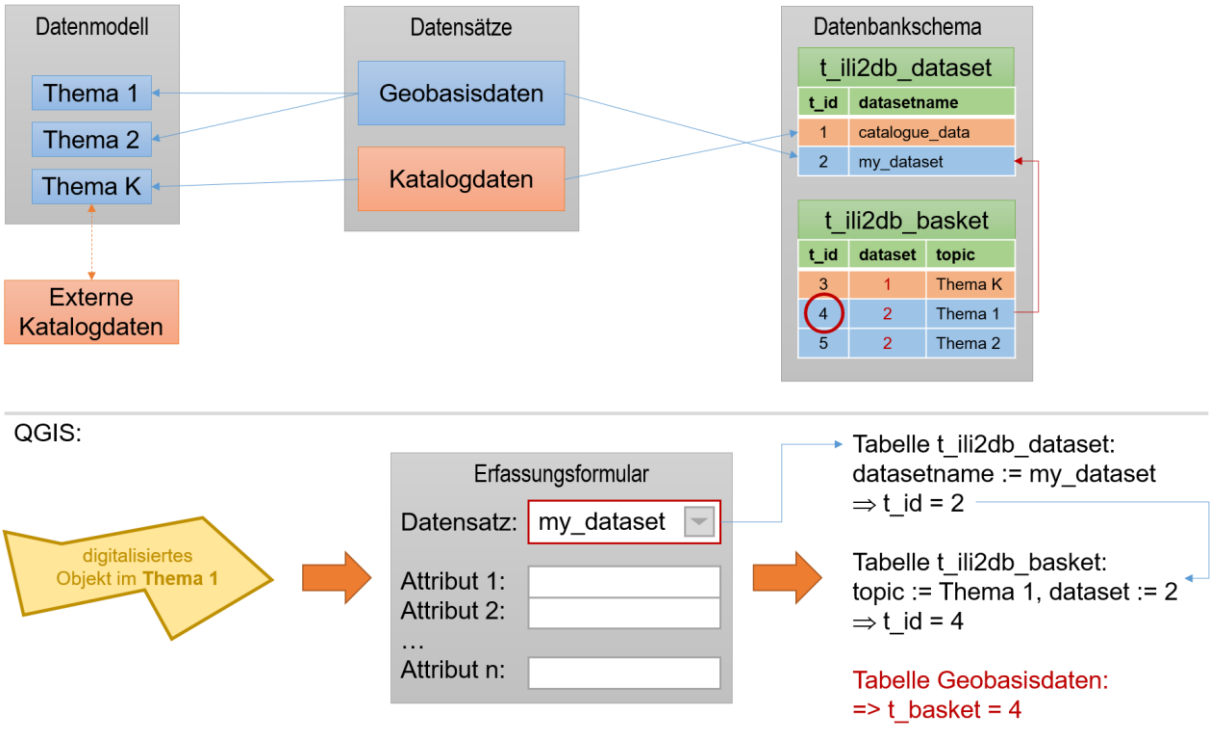
- Nach der Schemakonfiguration soll automatisch mindestens ein «allgemeiner» Datensatz mit dem/den entsprechenden Behälter/n erzeugt und in Verbindung gebracht werden.

- Nach der Schemakonfiguration sollen allfällige externe Kataloge automatisch gefunden und importiert werden. Der Benutzer soll dabei auf Wunsch selbst einen Datensatz-Namen definieren können. Es wird automatisch ein Behälter für die Katalogdaten erzeugt. Voraussetzung dafür ist die Registrierung von publizierten Katalogdaten in einem *Data Repository*.

- Bei jedem erstmaligen Datenimport soll der Benutzer auf Wunsch einen Datensatz-Namen definieren können. Grundsätzlich wird vom Programm ein eigener, generischer Name erzeugt und mit dem/den entsprechenden Behältern in Verbindung gebracht.

- Bei jedem Datenexport soll der Benutzer auf Wunsch auswählen können, welche Datensätze beziehungsweise welche Behälter exportiert werden sollen. Ohne Angabe werden alle Daten aus dem Schema exportiert, ausgenommen die Katalogdaten.

- Jede Tabelle, jeder «Layer» in einem mittels QGIS Model Baker generierten Projekt «weiss», zu welchem Behälter gemäss Datenmodell gehört. Nun könnte die Behälterreferenz (`t_basket`) automatisch ausgefüllt werden.
  Eigentlich ist es aber unschön, dem Benutzer diese technische Information aufzuzwingen. Viel anwenderfreundlicher wäre es, wenn in den Formularen immer der Name des Datensatzes exponiert würde, der Benutzer im Falle von mehreren Datensätzen den gewünschten Datensatz aus einer Auswahlliste auswählen könnte und dann beim Speichern im Hintergrund automatisch via Beziehung zwischen Behältern und Datensätzen die korrekte Behälterreferenz in das Feld `t_basket` gespeichert würde. Ein abstraktes Beispiel:



Der Benutzer soll also nicht selbst den korrekten Wert für `t_basket` erfassen müssen, sondern das Objekt dem gewünschten Datensatz zuweisen können.