ETH zürich

IKG – Institute of Cartography
and Geoinformation

Interoperability of GIS

Dr. Joram Schito
Lecturer | Spatial Data Scientist
29 October 2025, Zurich

geoinformation
engineering
ikg eth zurich

ETH zürich

IKG – Institute of Cartography
and Geoinformation

Interoperability

geoinformation
engineering
ikg eth zurich

Content

Learning objectives

Cognitive

- You can explain the basic concepts and the working mechanisms of interoperability and describe their advantages as well as their disadvantages.
- You can, by balancing pros and cons, evaluate the usefulness of data transfer between different systems by using different file formats
- You can describe advantages and disadvantages of making use of open source or non-open source applications.

Skills

- You can apply ogr2ogr for converting different geodata file formats into each other.
- You can import INTERLIS transfer files into a geodatabase by using the QGIS plugin “Model Baker.”
- You can use shell commands of ili2db libraries (in particular, ili2gpkg) to set up an INTERLIS model and to import several INTERLIS transfer files.

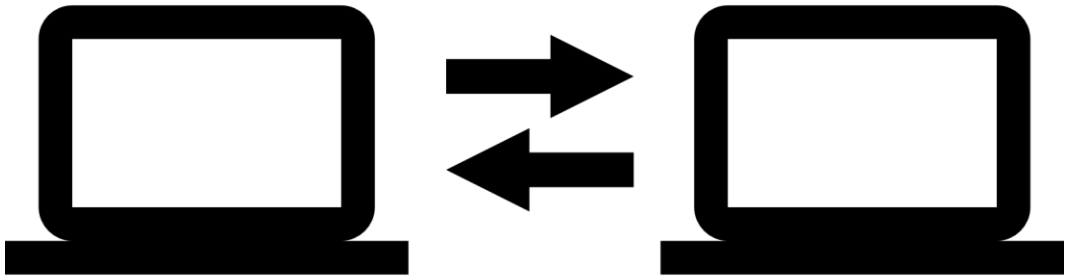
Learning objectives

Skills

- You can set up an application environment for your daily work with open source GIS tools.
- You can set up a GeoPackage (GPKG).
- You can convert geodata into different file formats by using QGIS or ogr2ogr.
- You can project geodata into different coordinate reference systems by using QGIS or ogr2ogr.
- You can handle the import/export pipeline of Model Baker when working with INTERLIS model and INTERLIS exchange data.
- You can interpret the content of INTERLIS transfer files correctly by combining all information provided by the data and by the INTERLIS model.
- You can create a concept of how you would implement an interoperable interface in an own GIS.
- You can describe advantages and disadvantages of current geodata formats and determine, which file format best fulfils a specific purpose.

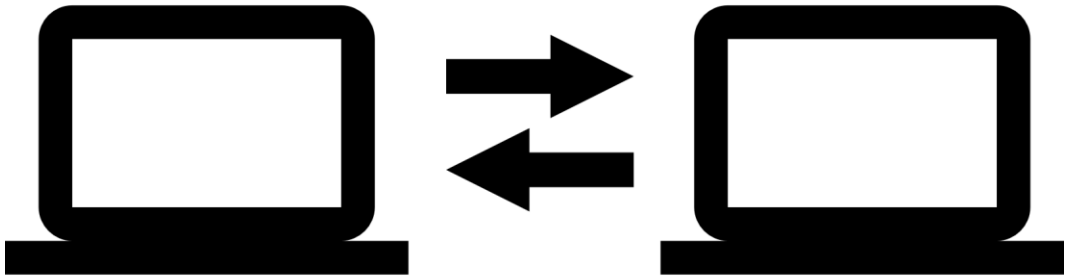
Interoperability principles

Why data transfer is important



- GIS need data from various sources and provide data to different clients.
- However, geodata:
  - are recorded at different places
  - are administrated with different systems
  - should be exchangeable between different systems
- Thus, an efficient data transfer mechanism is crucial for the work with geodata.

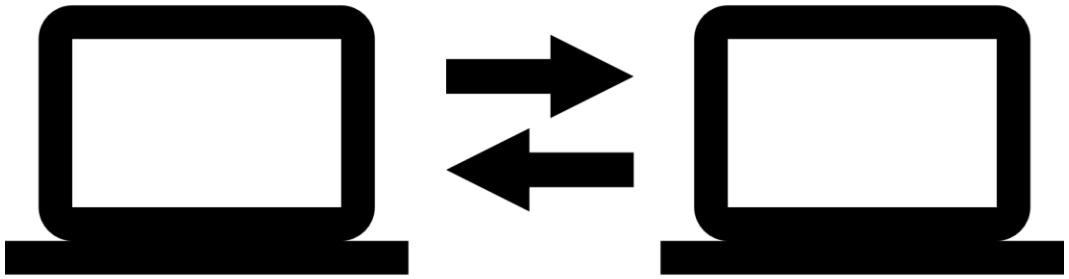
The dilemma of data transfer



- The representation might depend on the topic
  - **Different data models**
- Each data model can be stored in different formats
  - **Different file formats**
- The content of the geodata (including metadata) might vary across different data sets
  - **Different attribute structures**
- Different GIS might have different requirements and preferences
  - **Different data structures**

Task: How many bidirectional drivers would we need to convert each of n file formats into the others? (2 min)

The dilemma of data transfer

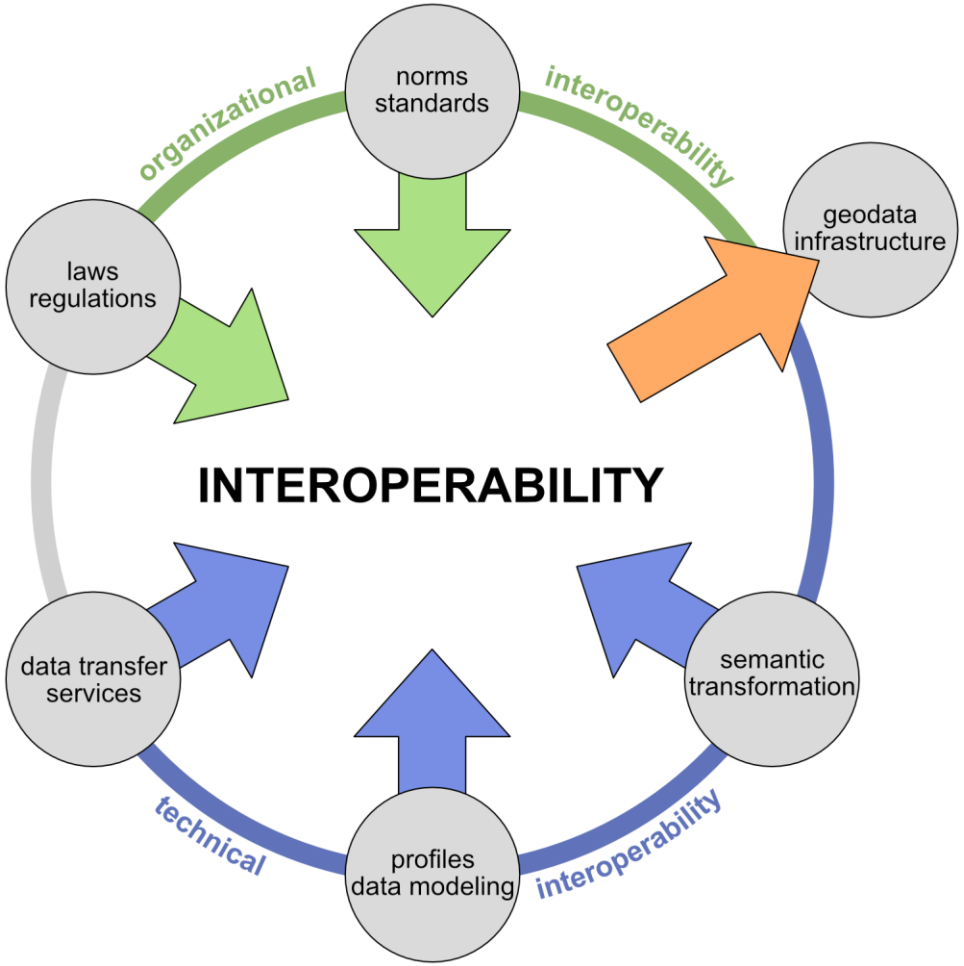


- The representation might depend on the topic
  - **Different data models**
- Each data model can be stored in different formats
  - **Different file formats**
- The content of the geodata (including metadata) might vary across different data sets
  - **Different attribute structures**
- Different GIS might have different requirements and preferences
  - **Different data structures**

Task: How many bidirectional drivers would we need to convert each of n file formats into the others? (2 min)

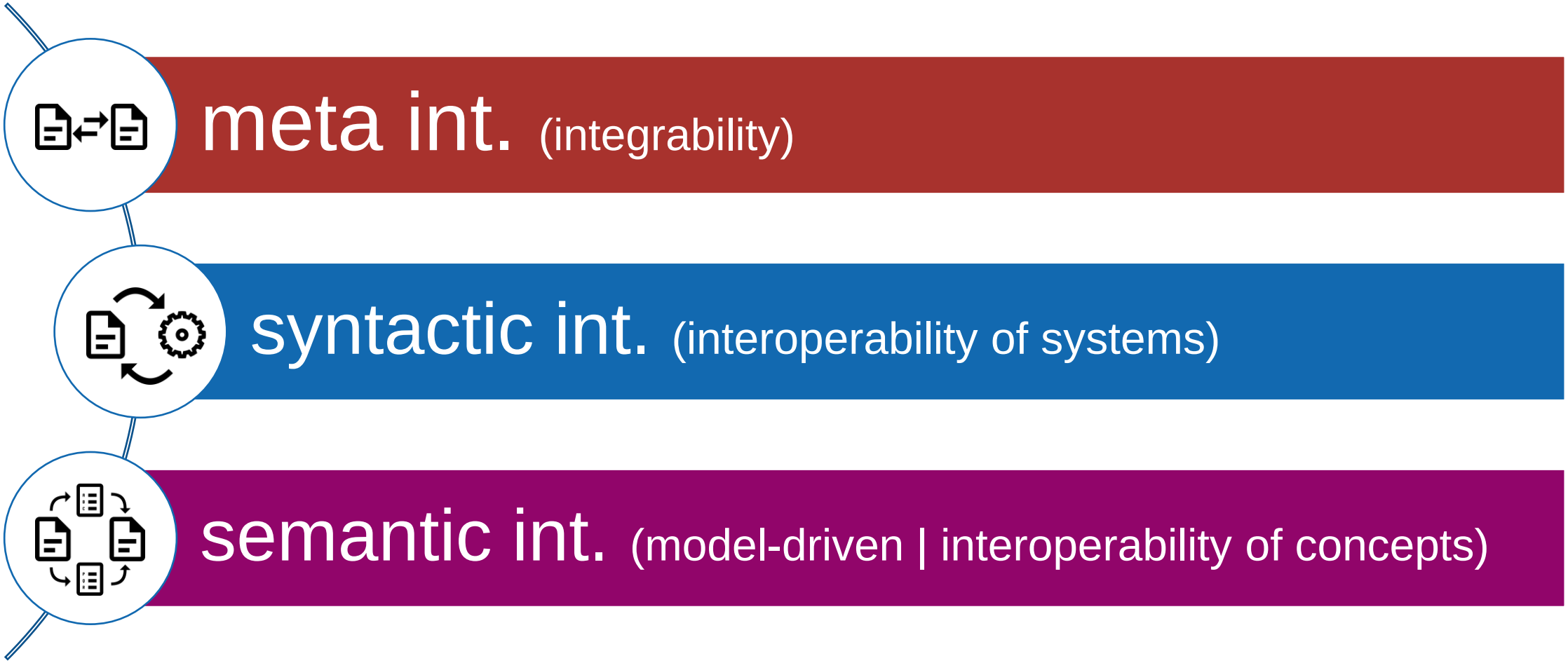
We would need $\frac{n \cdot (n-1)}{2}$ bidirectional drivers.

Interoperability principles

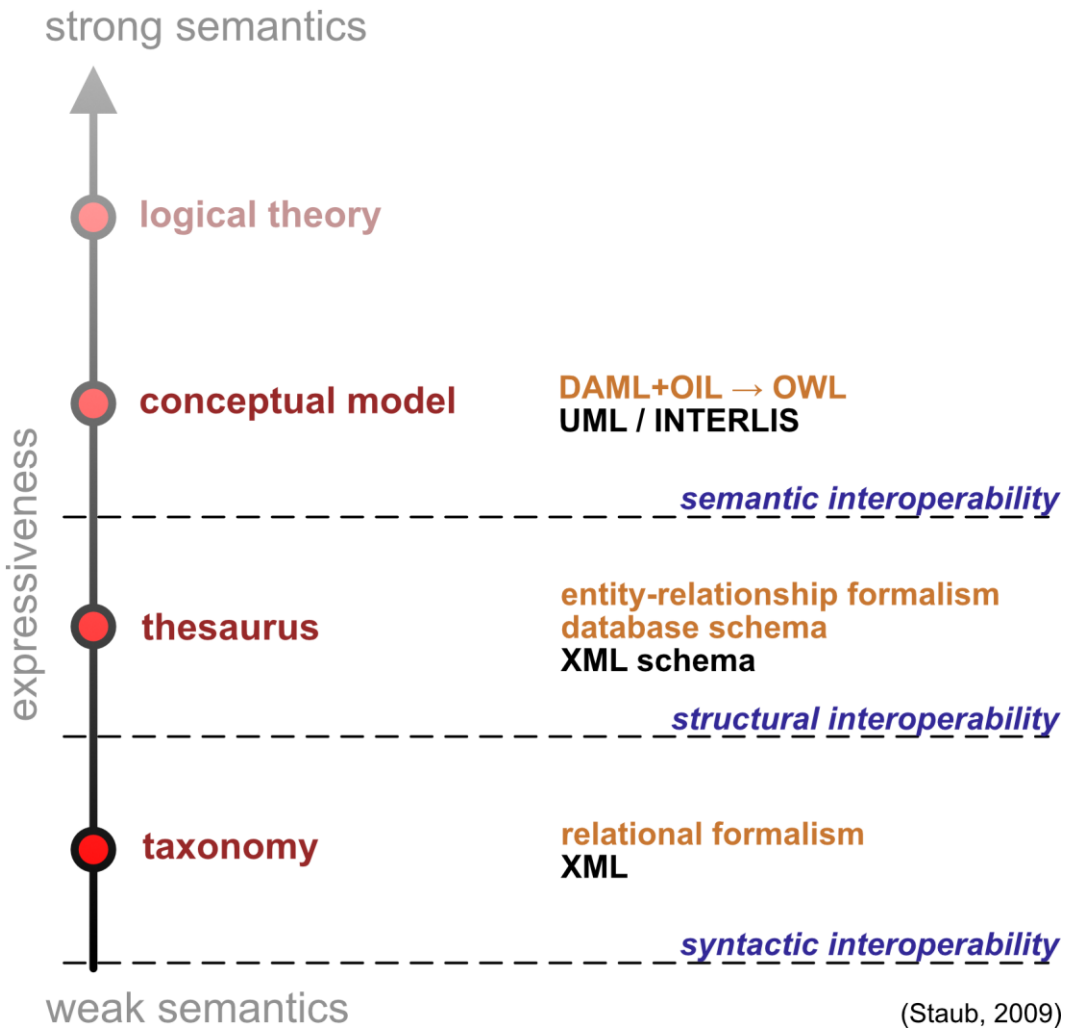


- Definition of Straub, P. (2009): Interoperability is the capability of Geographic Information Systems to work together.
- The characteristics of interoperability can be subdivided into organizational and technical aspects.
- Norms, standards, and regulations build the **organizational** basis for interoperability.
- Profiles, data models, data transfer, services, and semantic transformation allow interoperability between different GIS on a **technical** level.

Principles of technical interoperability



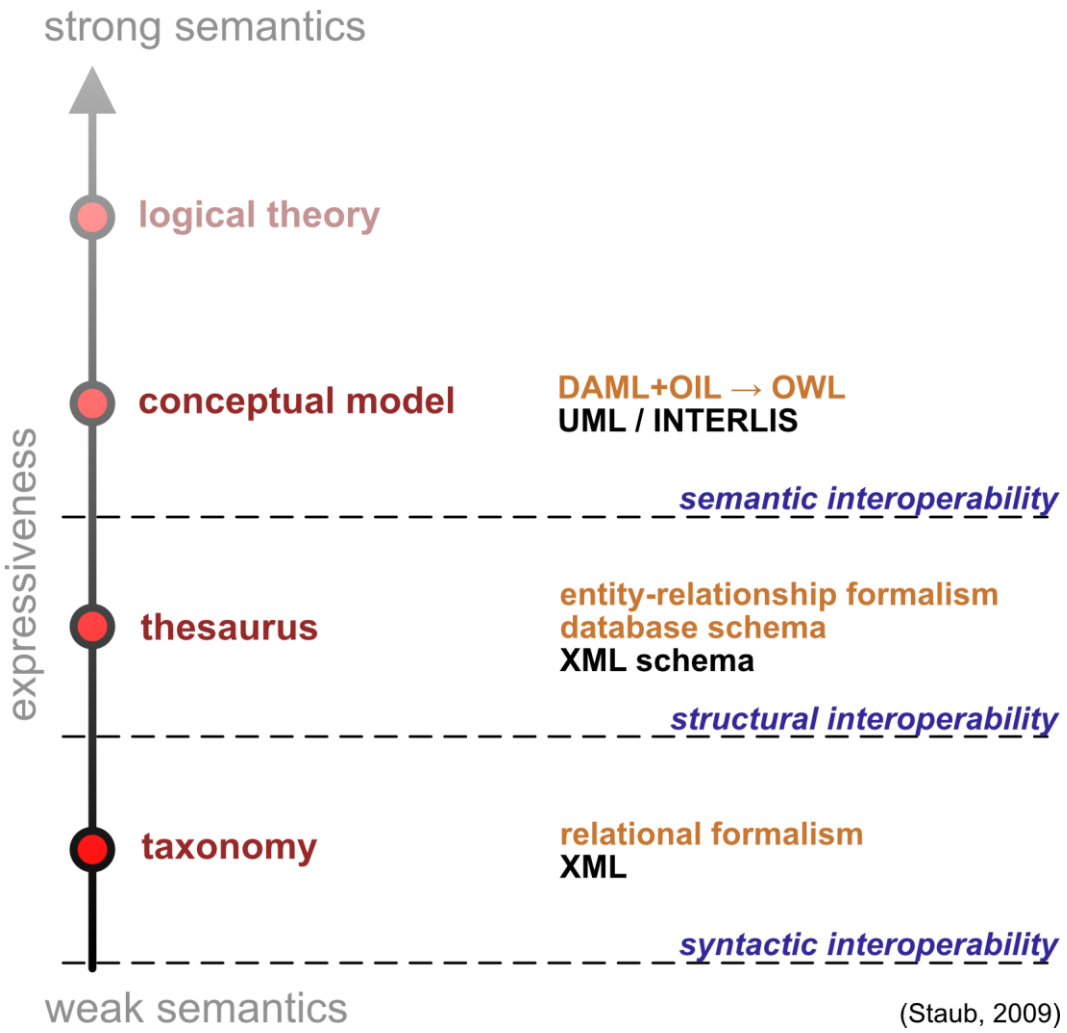
Ontology spectrum



Ontologies can be interpreted as a spectrum from a taxonomy to logic theory.

- **Taxonomy**, a scheme of classification, especially a hierarchical classification, in which things are organized into groups or types.
- **Thesaurus**, a synonym dictionary that arranges words by their meanings, aiming at describing and representing a topic. It can be arranged by a hierarchy of broader and narrower terms or as lists of synonyms and antonyms.
- **Conceptual model**, a simplified and abstract representation of a topic that represents how those parts relate to each other.
- **Logic theory**, a formal system used to represent and analyze the meaning of statements, sentences, or propositions in a structured way.

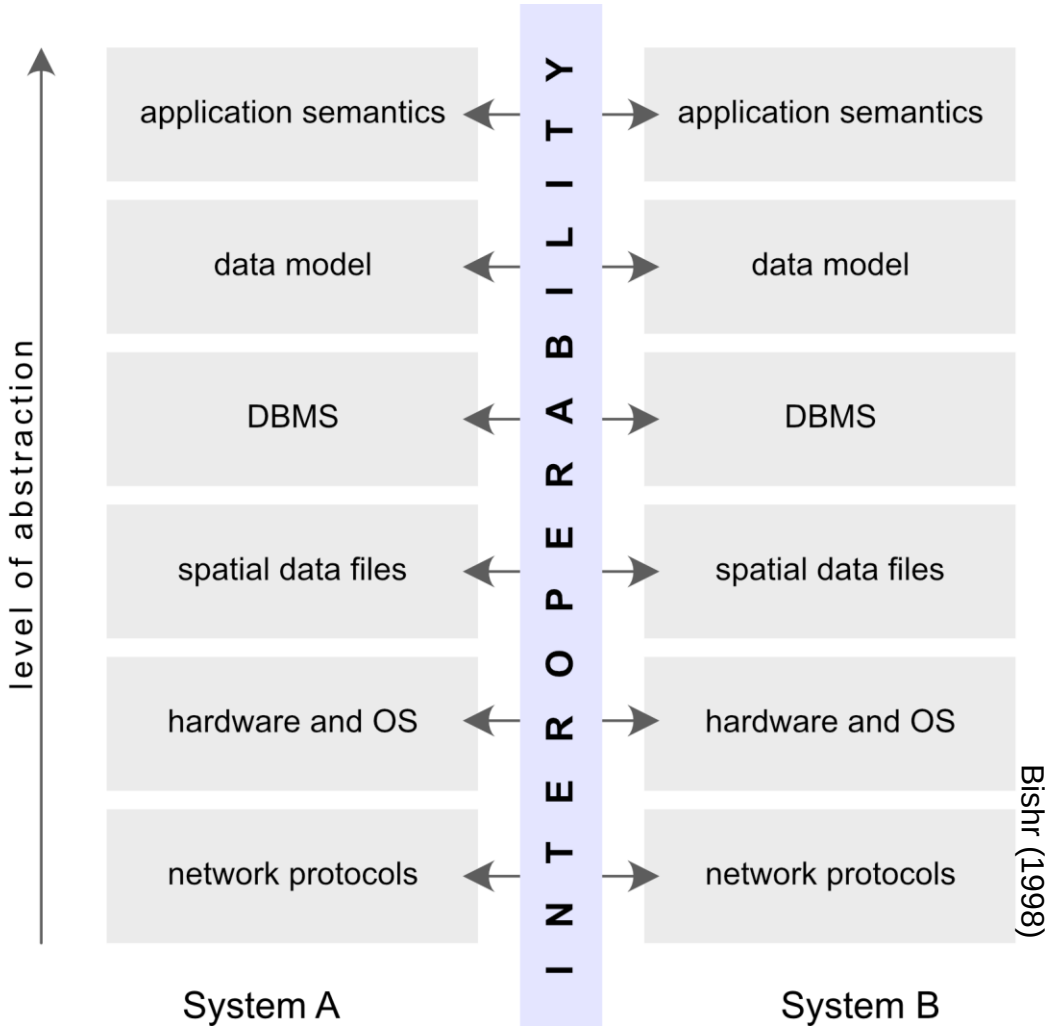
Ontology spectrum



Each of the components contributes to the expressiveness of ontologies, whose levels are:

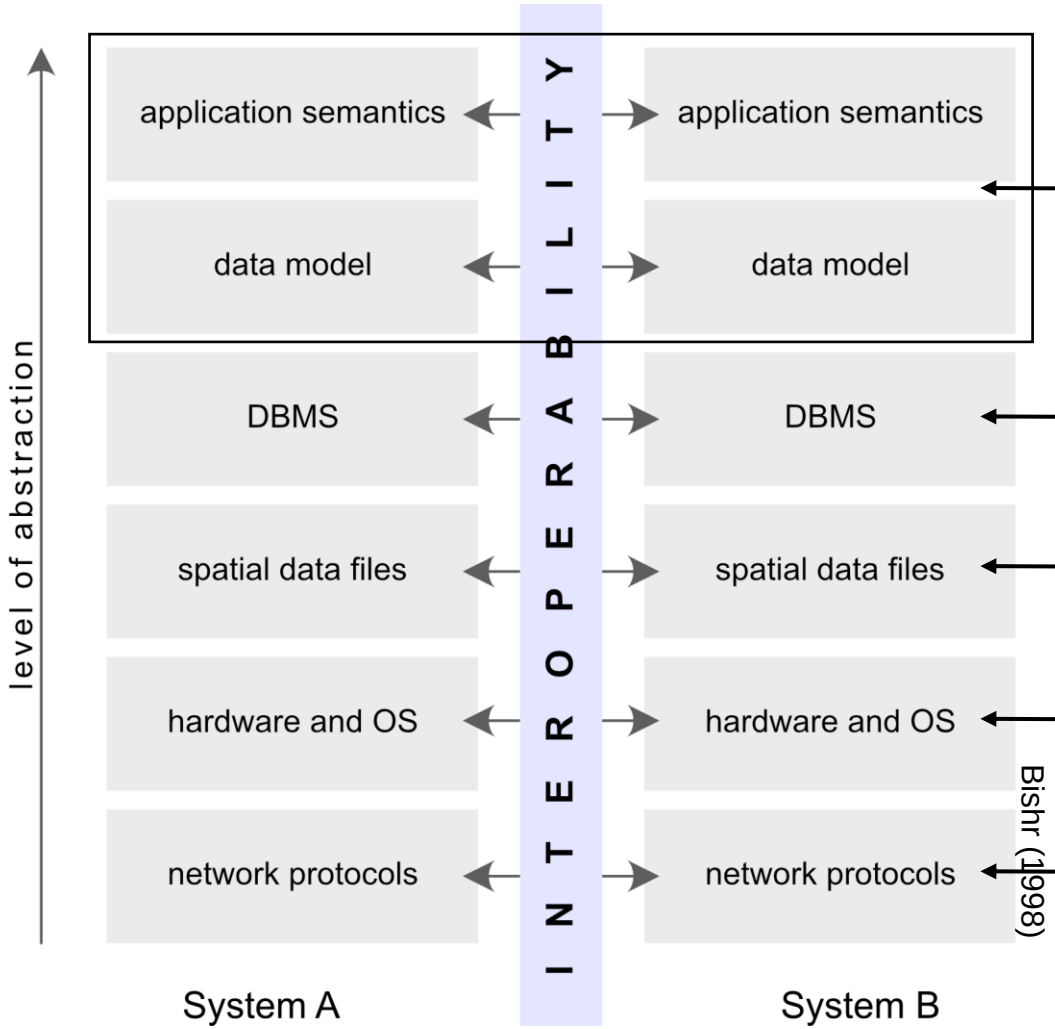
- Lowest Level: Syntactic interoperability based on standardized exchange format (e.g., XML).
- Intermediate Level: Structural interoperability achieved by mapping schemas.
- Highest Level: Semantic interoperability attained through abstract formalisms and conceptual data models.

Implementation levels of interoperability



- According to Bishr (1998), interoperability between two GIS takes place at different levels of hardware abstraction.
- Thus, building an interoperable GIS requires to address all technological aspects that affect the used technology.

Implementation levels of interoperability



INTERLIS

Interoperable per se as geodata is stored and processed server-side

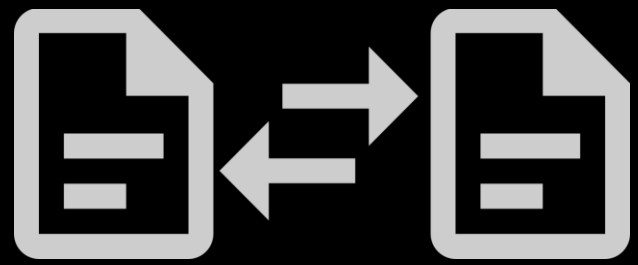
Many different: GeoJSON, KML, …

Platform-independent software / Web-GIS / web-based platforms

OGC web services (WMS/WMTS/WFS/...)

Interoperability principles

meta interoperability



What interoperability is not

No joke—I recently saw this request:

- “Please provide geodata about your utilities on any topic you like, in any format.”
- What that means in practice:
  - No common topic = no semantic logic
  - No common data model
  - No common quality standards
  - No common attributes
  - No common attributes types
  - No common encoding
  - No common CRS
  - No common file format
- At best, this minimal level of agreement would be considered **integrability**.

How to ensure integrability

The specification of a **profile** is key for integrability. A profile is an application of norms and standards which restricts or extends a given specification while retaining the compulsory core elements. Profiles allow users to customize a standard to their specific needs. A profile encompasses:

- Data encoding and file format
- geospatial web services and their capabilities
- Coordinate reference system (CRS)
- Metadata standards
- Conceptual model (including field types), derived from a universe of discourse.
- Temporal profiles (representation of time zones)
- Semantic profiles (define specific meanings)

How to ensure integrability

```
# Level 0
if i['geometry']['type'] == 'Point':
    # print("in level 0")

    # Special handling for points
    coordinates = [coordinates]

    # Retrieve the x and y coordinate and if it exists, also the z coordinate, transform them and
    # append them to the same list structure as we iterated through
    for (x2, y2, *z2) in transformer.itransform(coordinates):
        record = (x2, y2, *z2)

# Level 1
elif i['geometry']['type'] in {'LineString' or 'MultiPoint'}:
    # print("in level 1")
    for (x2, y2, *z2) in transformer.itransform(coordinates):
        record.append((x2, y2, *z2))

# Level 2
elif str(i['geometry']['type']) in {'Polygon', 'MultiLineString'}:
    # print("in level 2")
    for level_1 in coordinates:
        level_2_list = []
        for (x2, y2, *z2) in transformer.itransform(level_1):
            level_2_list.append((x2, y2, *z2))
        record.append(level_2_list)

# Level 3
elif i['geometry']['type'] == 'MultiPolygon':
    # print("in level 3")
    for level_1 in coordinates:
        level_2_list = []
        for level_2 in level_1:
            level_3_list = []
            for (x2, y2, *z2) in transformer.itransform(level_2):
                level_3_list.append((x2, y2, *z2))
            level_2_list.append(level_3_list)
        record.append(level_2_list)
```

After having agreed on everything, you can establish mechanisms in your code that ensure that the data is processed by following the defined specifications. Examples:

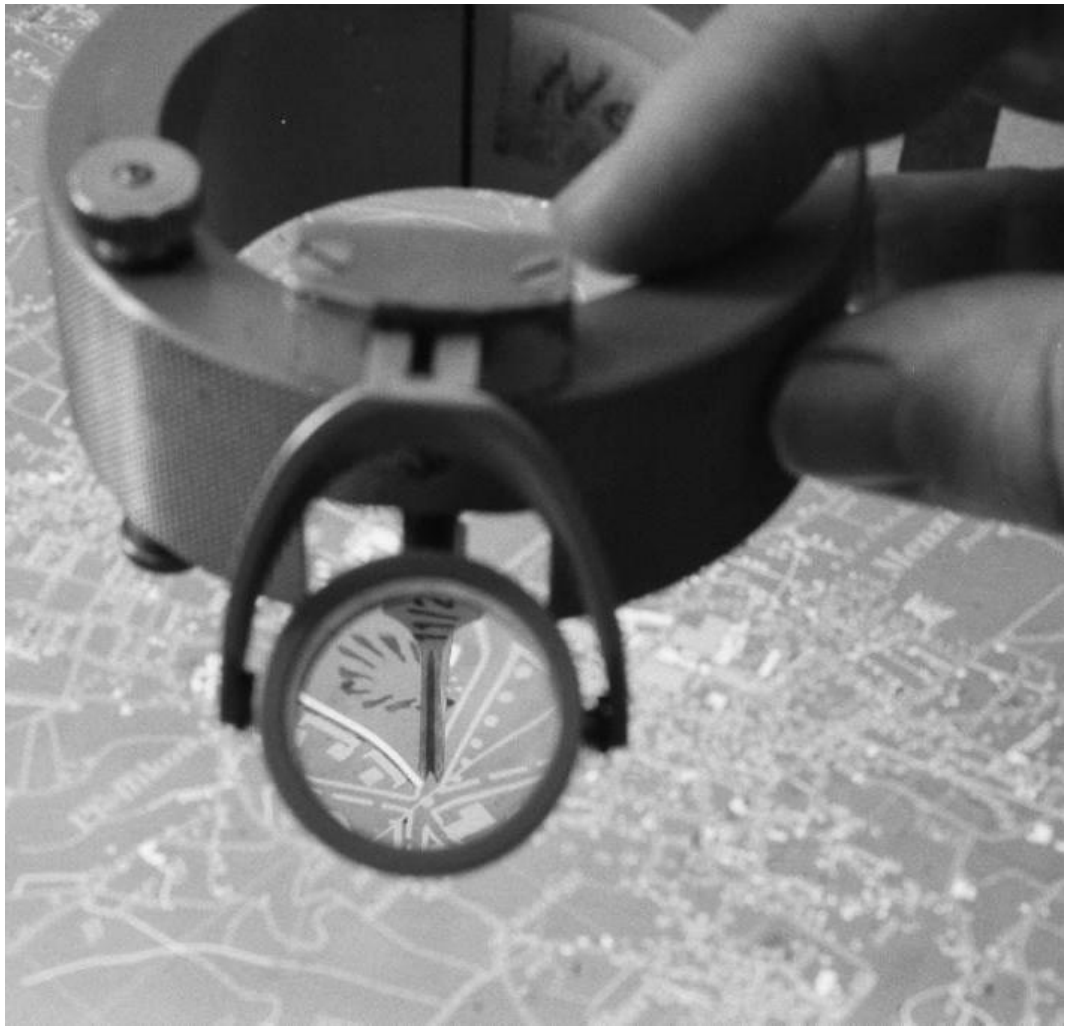
- Your algorithm interprets different projection objects equally, such as PROJ.4, WKT, or by entering “4326”.
- Your algorithm converts all coordinates to the projected CRS EPSG:2056.
- Your algorithm can read both, POINT or MULTIPOINT while a commonly defined geometry object is used.
- MSSQL recognizes date formats automatically when using TO_DATE(___).
- ogr2ogr converts file format A to file format B.

Some remarks on different geodata file formats

- Specify both, the start and the target file format and the referred data model very clearly.
- A file-based data integrability mechanism can be used to exchange data between two GIS with different proprietary file formats by using an open standard format as intermediator.
- Pay attention to use file formats that keep the geographical characteristics of geodata (e.g., GeoJSON instead of JSON).

- **Question: Have you ever used the following tools for converting geodata files?**

Mechanism and limitations of the format-based transfer



© ETH Zurich, see interesting article

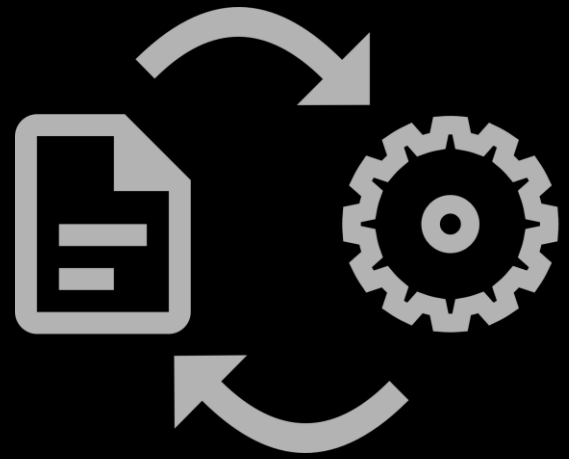
Task (In groups of 2-3 people, 5 min):

What do you think about a fixed standard transfer format with an identical data structure across all cantons for official surveying? Would there be any limitations?

The requirements regarding the attribute structure and the geodata that are collected might differ across the cantons (or even countries for international collaboration projects). A fixed standard transfer format might be too rigid and inflexible.

Interoperability principles

syntactic interoperability



Syntactic interoperability

Midjourney prompt: “imagine syntactic interoperability”

- Syntactic interoperability means that available data can be used through a standardized service for using, manipulating, retrieving, and visualizing geodata.
- Such a services uses:
  - standardized communication protocols (including commands) and
  - standardized data transfer formats.
- In practice, syntactic interoperability is enabled by the standardized **OGC web services**.

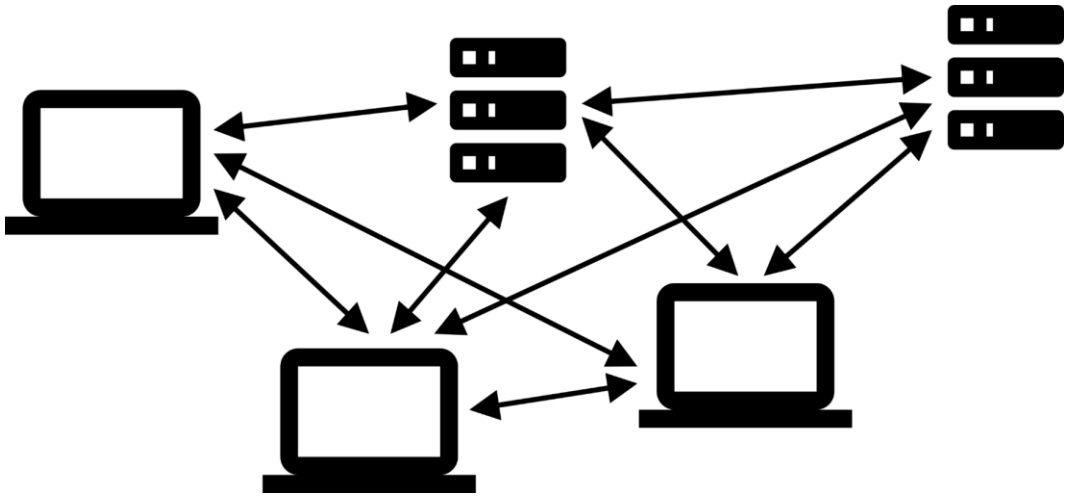
Syntactic interoperability

- Pro: Useful for various evaluations, such as:
  - geodata visualization
  - list of few selected objects
  - displaying calculation/query results
- Pro: OGC services are platform-neutral; GIS systems don’t need to understand each other’s data management.
- Pro: Users receive only the requested data in a standard format, aiding integration.
- Pro: OGC standards are widely supported across GIS tools.
- Pro: Services can be extended, but extended profiles are not standardized.
- Con: Data structures are hidden, so the data may not align with the user's schema.

Syntactic interoperability

- Despite not being part of the standard OGC web services, researchers implemented various useful extensions, such as:
  - Geoprocessing workflows (Percivall, 2010)
  - GIS-MCDA mechanisms (Jelokhani et al., 2018)
  - Methods that allow semantic transformations like mdWFS (Donaubauer et al., 2007) and UMLT (Staub, 2007), even though they have not become a standard yet.

Advantages and limitations of interoperability in GIS



Task (group discussion, 2–3 min):

Who of you did ever use OGC Web Services? Did the services provide you the results you needed? Which limitations did you encounter?

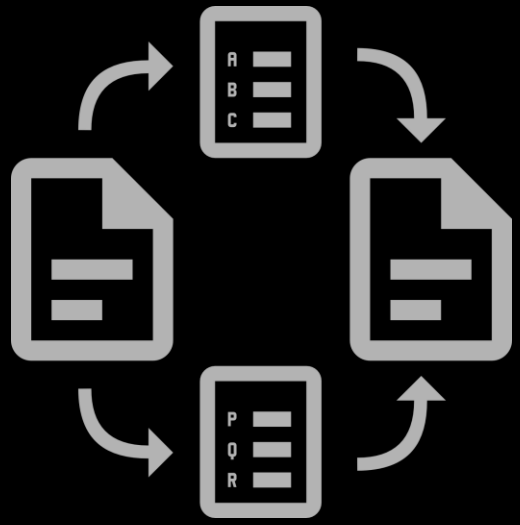
The most popular OGC Web Services are WMTS, WCS, and WFS, thus, retrieving, filtering, and visualizing geodata.

OGC Web Services offer a standardized interface for using available data. Thus, complex methods for uploading own data, retrieving foreign data, and using the services for more complex geodata manipulation processes, are often not available.

Moreover, the services may not provide the required data or information for further analyses.

Interoperability principles

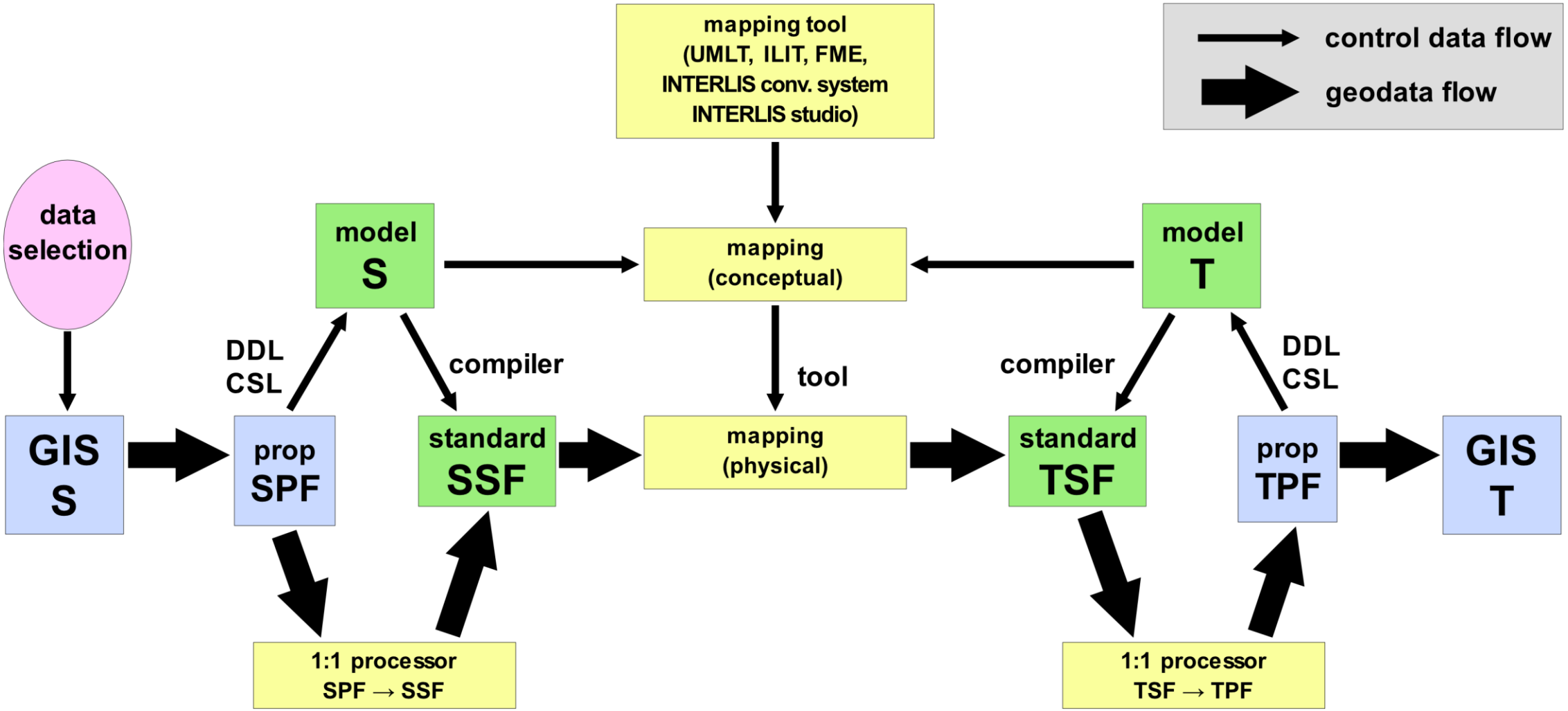
semantic (model-driven) interoperability



Model-driven data transfer with restructuring (MDDTWR) – main principle (simplified)

- Problem: Data should be exchanged between two GIS. However, the used data models may be defined differently.
- Procedure:
  1. The user selects the data in the start GIS
  2. The start GIS exports the description of the model and the data
  3. The mapping functionality of the target GIS allows structure conversion (**restructuring**)
  4. The target GIS imports and interprets the model
  5. The target GIS can read the restructured data

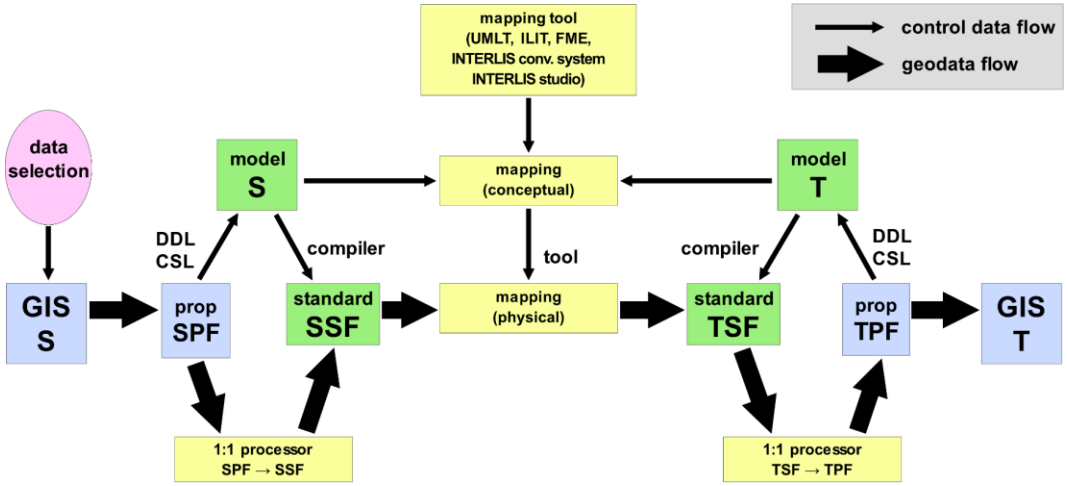
Model-driven data transfer with restructuring: main principle



Model-driven data transfer with restructuring: main principle

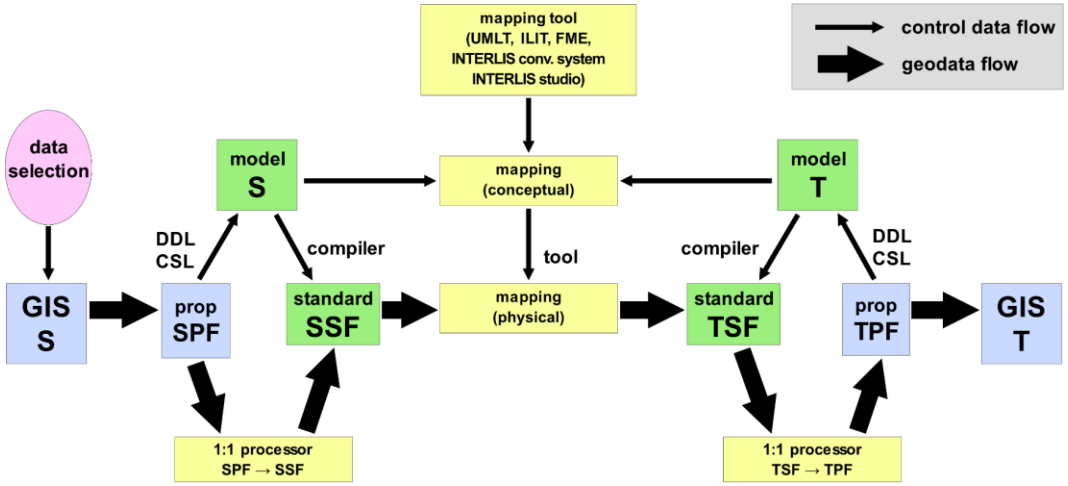
1. Select the data of your choice in the start GIS.
2. Export the transfer file to the proprietary GIS format SPF. Then, describe it in natural language.
3. Precisely describe the data structure of the SPF file by a DDL/CSL → model S.
4. The compiler provides the description of the standard format SSF according to the model S.
5. Describe the expected TPF (written in a proprietary format) in natural language.
6. Precisely describe the data structure of the file TPF by a DDL/CSL → model T.
7. Program the one-to-one processor to reformat SPF → SSF.
8. Restructure the model S → model T while the file transformation SSF → TSF is made automatically.
9. Compiler provides the description of the standard format TSF according to the model T.
10. Program the one-to-one processor to reformat TSF → TPF.

Model-driven data transfer with restructuring: main principle



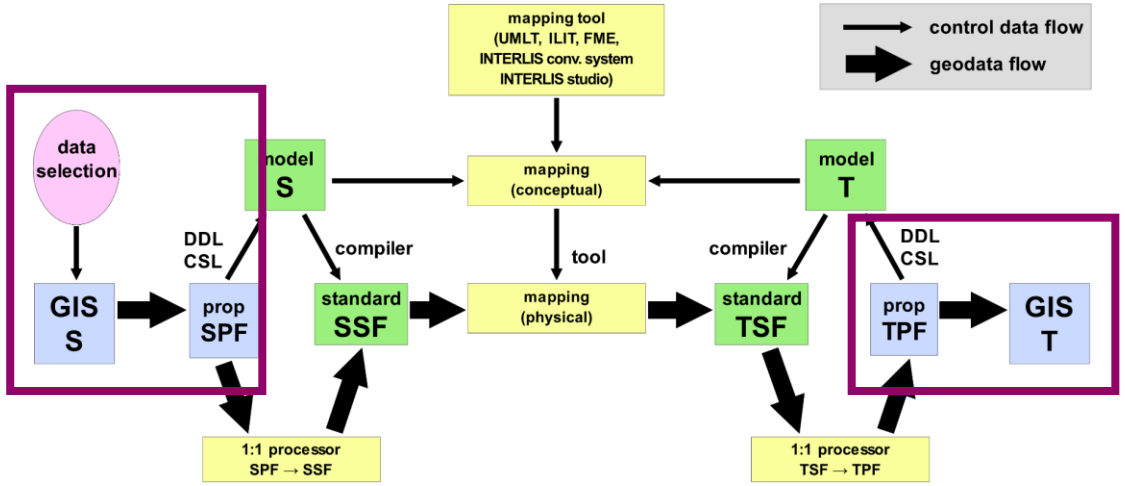
- The transfer of flexible data schemas can be achieved by:
  - describing **two data models** on a conceptual level: one of the data that should be transferred and one the target data
  - describing the rules according to which the data is mapped (**mapping rules**)
- By this, the target standard format can automatically be derived from the incoming data, the target data model, and the mapping rules.
- Requirements:
  - standardized conceptual data description language (DDL)
  - standardized procedure (compiler) to derive the transfer format from the data model

Model-driven data transfer with restructuring: overview



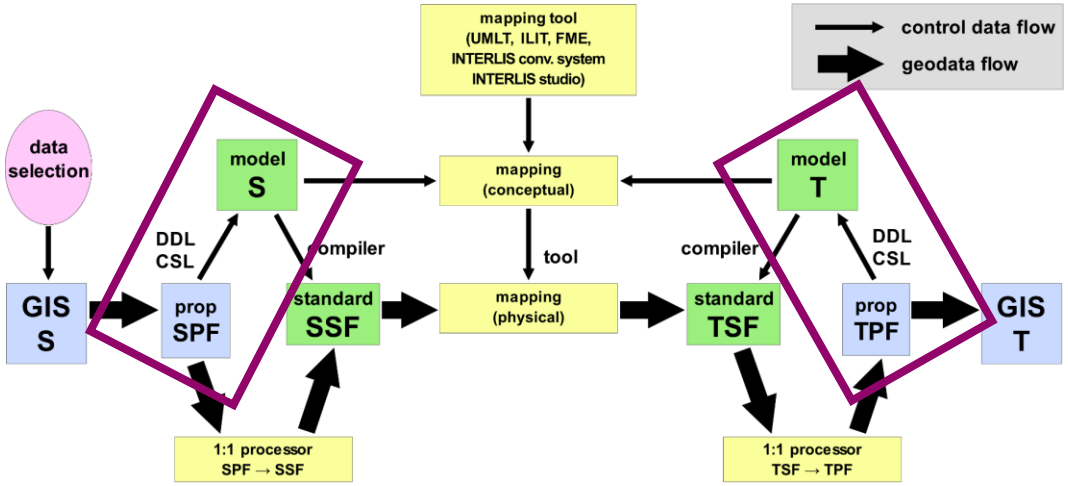
- A GIS with a model-driven interface simplifies the data transfer with a flexible data format.
- A mapping functionality in the target GIS makes restructuring simple.
- The model-driven approach with restructuring is currently seen as the apex method for enabling interoperability with GIS. However, research in this field was mainly conducted in the German-speaking region in Europe between 2006–2011.
- Restructuring needs human knowledge and interaction (semantics) to match the attributes to each other. Thus, data transfer with restructuring cannot be done automatically yet, but approaches from Machine Learning are promising for doing so in the future.

A) Description of the universe of discourse (in natural language)



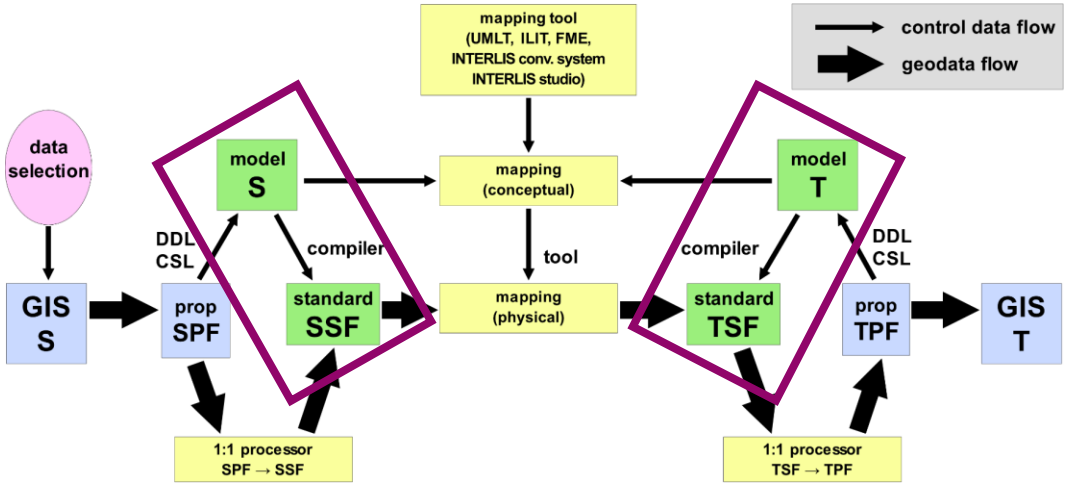
- Define the **universe of discourse**, which describes the reality as precisely as possible by using **natural language**.
- Natural language comprise documents that contain everything that describe the reality, which can encompass tables, technical documentations, object type catalogs, maps, etc.
- With the model-driven data transfer, the universe of discourse has to be described for the start as well as for the target system.

B) Conceptual and logic data modeling (UML, INTERLIS)



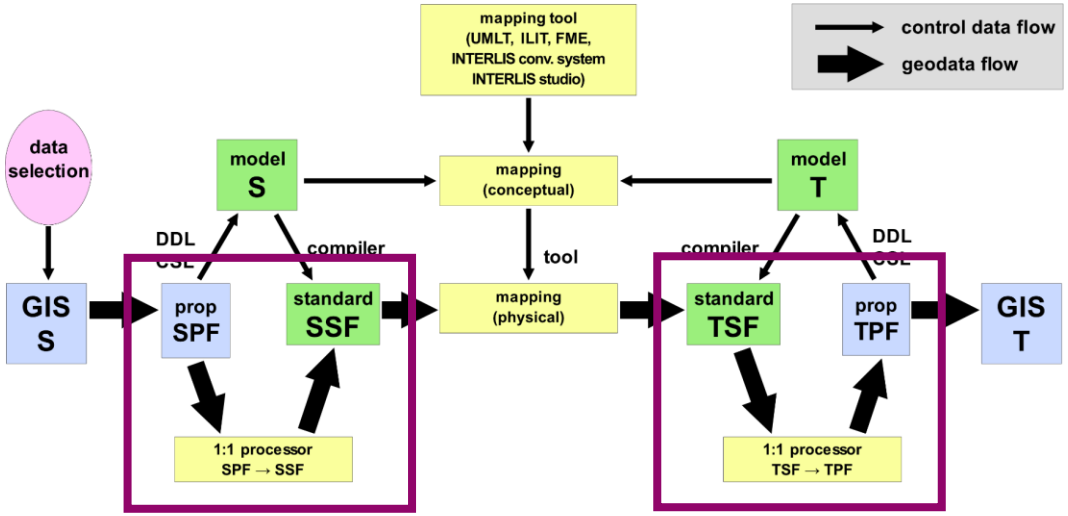
- Formulate the **data structure** of the proprietary transfer format **graphically and/or textually**
- For doing so, use a **conceptual schema language** (CSL) or a **data description language** (DDL)
  - graphically with UML
  - textually with INTERLIS 2
- Describe classes, attributes, relationships, units, consistency constraints, etc. as precisely as possible

C) Description of the (model-specific) standard transfer format



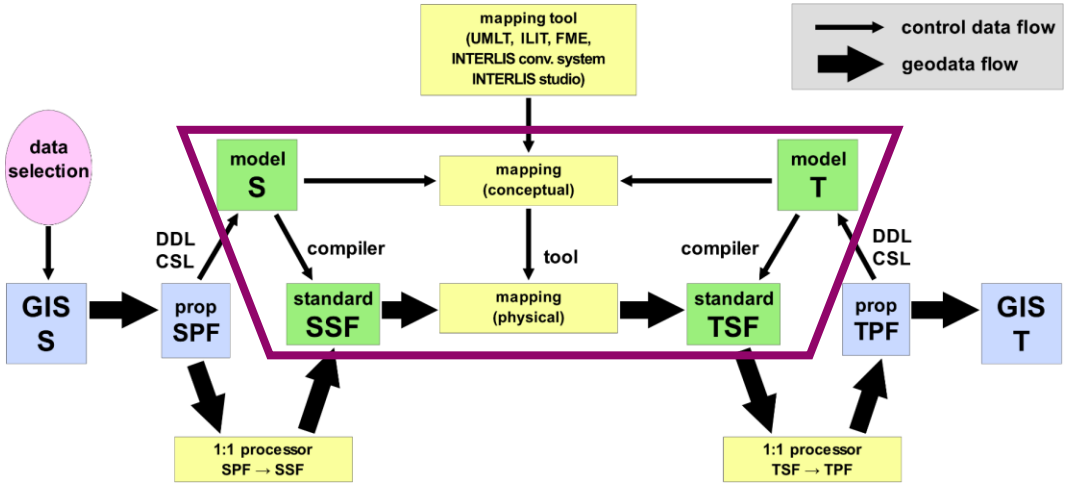
- The standard transfer file is physically created by the one-to-one processor while referring to the rules defined in the conceptual data model.
- It refers to a model and that contains geometric and thematic information.
  - INTERLIS 1 transfer format: ITF
  - INTERLIS 2 transfer format: XTF
- The compiler derives the format description (the structure including the attribute names and types) from the transfer file.

D) One-to-one processor



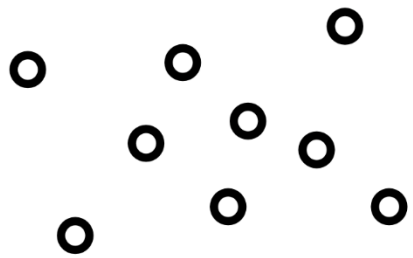
- The one-to-one processor reformats data from the proprietary format into the corresponding standard format and vice versa.
- Usually, it is a script of any programming language that applies specific rules for recognizing and re-formatting character patterns.
- The file in the proprietary format and the file in the standard format correspond to the same data structure. Therefore the name one-to-one processor.

E) Semantic transformation

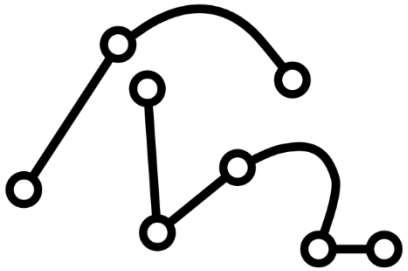


- Applies a **semantic transformation** to convert the data structure of the start system to the data structure of the target system.
- The conversion of the data structure is defined on a **conceptual level**
- If the model conversion is defined and if the start data are present in the standard format, then these data are automatically transformed into the standard format of the target system.
- Mapping rules between conceptual data models encompass actions and activities, which can be described formally by using UMLT.
- UMLT fosters **interoperability in a service-oriented architecture (SOA)**, which enables platform-independent geodata processing.

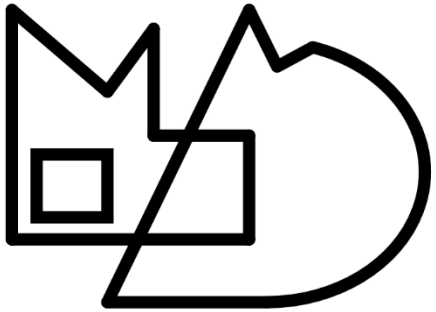
Model-driven data transfer with restructuring (MDDTWR) – conclusions



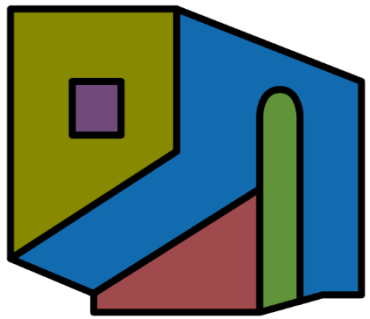
COORD



POLYLINE



SURFACE



AREA

- The model-based transfer mechanism allows:
  - data transfer across different GIS
  - flexible data format since it is automatically generated out of the data model
- Limitations:
  - the GIS needs a model-based interface
  - agreement to use the same standardized data description language
- The MDDTWR can be established by using the DDL **INTERLIS**. To date, INTERLIS is the only solution for conveying the model-driven transfer mechanism that can handle geodata.

Plugin “Model Baker” for QGIS

Install QGIS and Model Baker

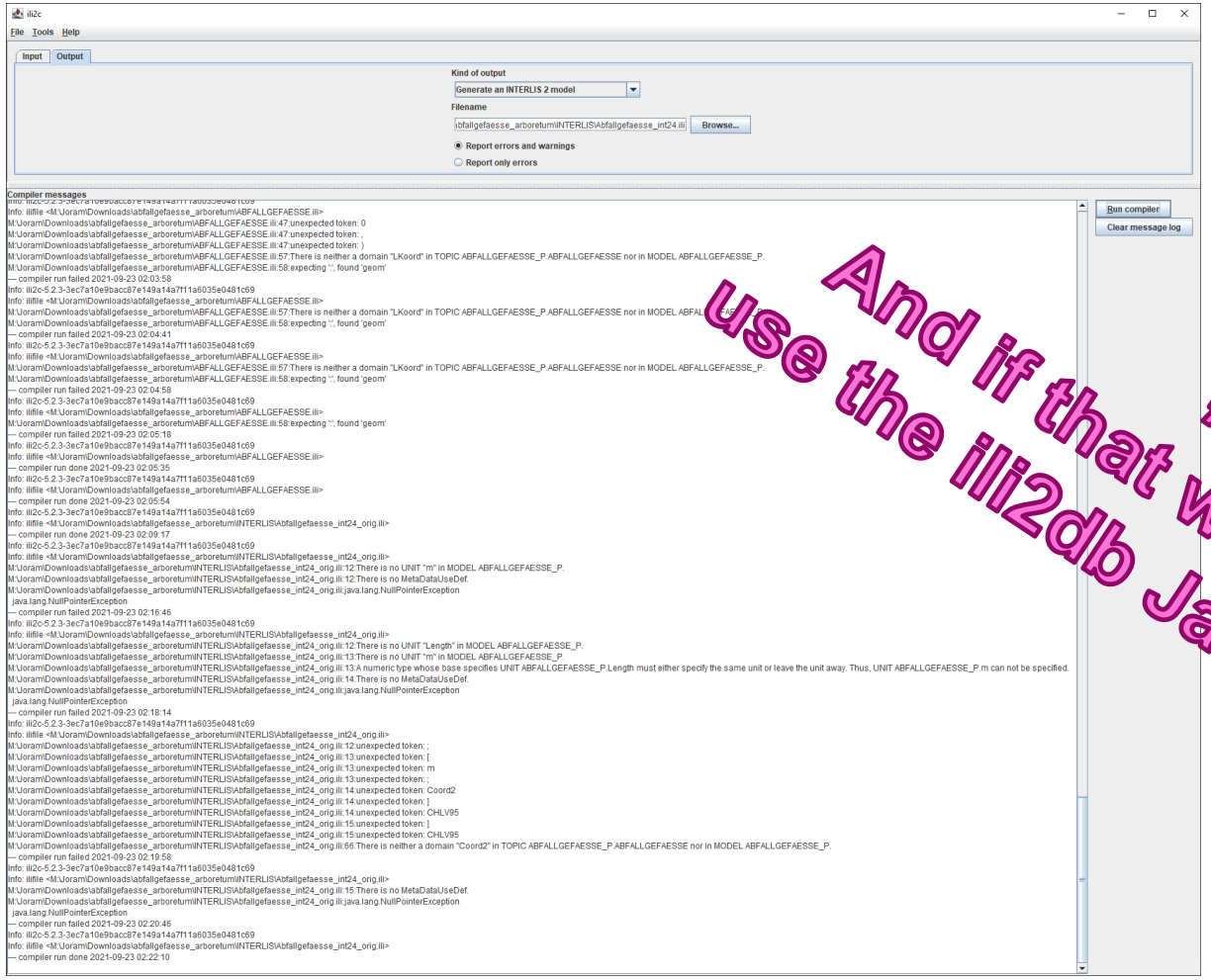
Prerequisites

- Since Model Baker requires Java JRE 8, we recommend you install **`Amazon Corretto 8`** (see this website).

Procedure

1. Install QGIS
2. Make sure that you are connected to the internet.
3. Open the plugin manager. Search and install the plugin called «Model Baker» (developed by OPENGIS.ch).
4. You can decide whether you want to work either with PostGIS/PostgreSQL databases or GPKG.
5. Do the steps 1–6 on the following slides.

Using Model Baker



And if that wouldn't work, use the ili2db Java application.

###!!!

The library ili2db

ili2db

INTERLIS 2-loader for databases

ili2pg, ili2gpkg and ili2fgdb are programs that write an INTERLIS transfer file according to an INTERLIS model into a database (PostgrSQL/PostGIS, GeoPackage or ESRI FileGDB) or create such a transfer file from a database.

Software errors of ili2db or new tool requirements can be registered directly on the developer platform GitHub.

- ili2pg (for PostgreSQL/PostGIS)
- ili2gpkg (for GeoPackage)
- ili2fgdb (for ESRI FileGDB)

Download **ili2pg** – Version 5.0.1 of 25.09.2023, ZIP

Download **ili2gpkg** – Version 5.0.1 of 25.09.2023, ZIP

Download **ili2fgdb** – Version 5.0.1 of 25.09.2023, ZIP

- The library ili2db is the parent library of ili2pg, ili2gpkg, and ili2fgdb.
- Since Model Baker supports both, ili2pg and ili2gpkg, you can use either a PostgreSQL database or a Geopackage for saving the data.

Exercise Series

Handling INTERLIS data by using Model Baker, ili2db, and SQL

ETH zürich

Exercises

geoinformation engineering ikg eth zurich

Protecting cultural goods accurately is important…

france24.fr, 19.10.2025

lesechos.fr

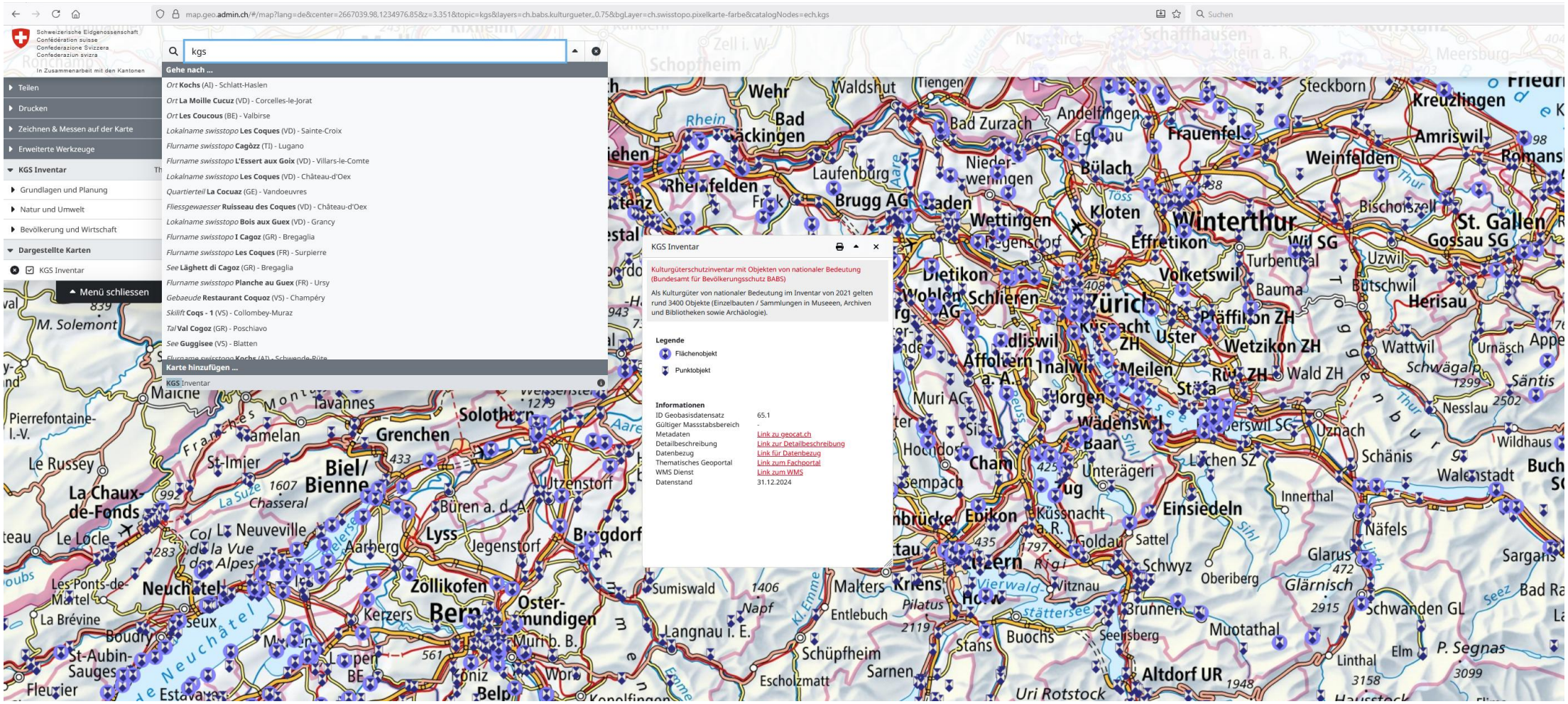
Goal: Import the KGS and ISOS data set to a database and merge them into an unified data set

KGS: import by Model Baker

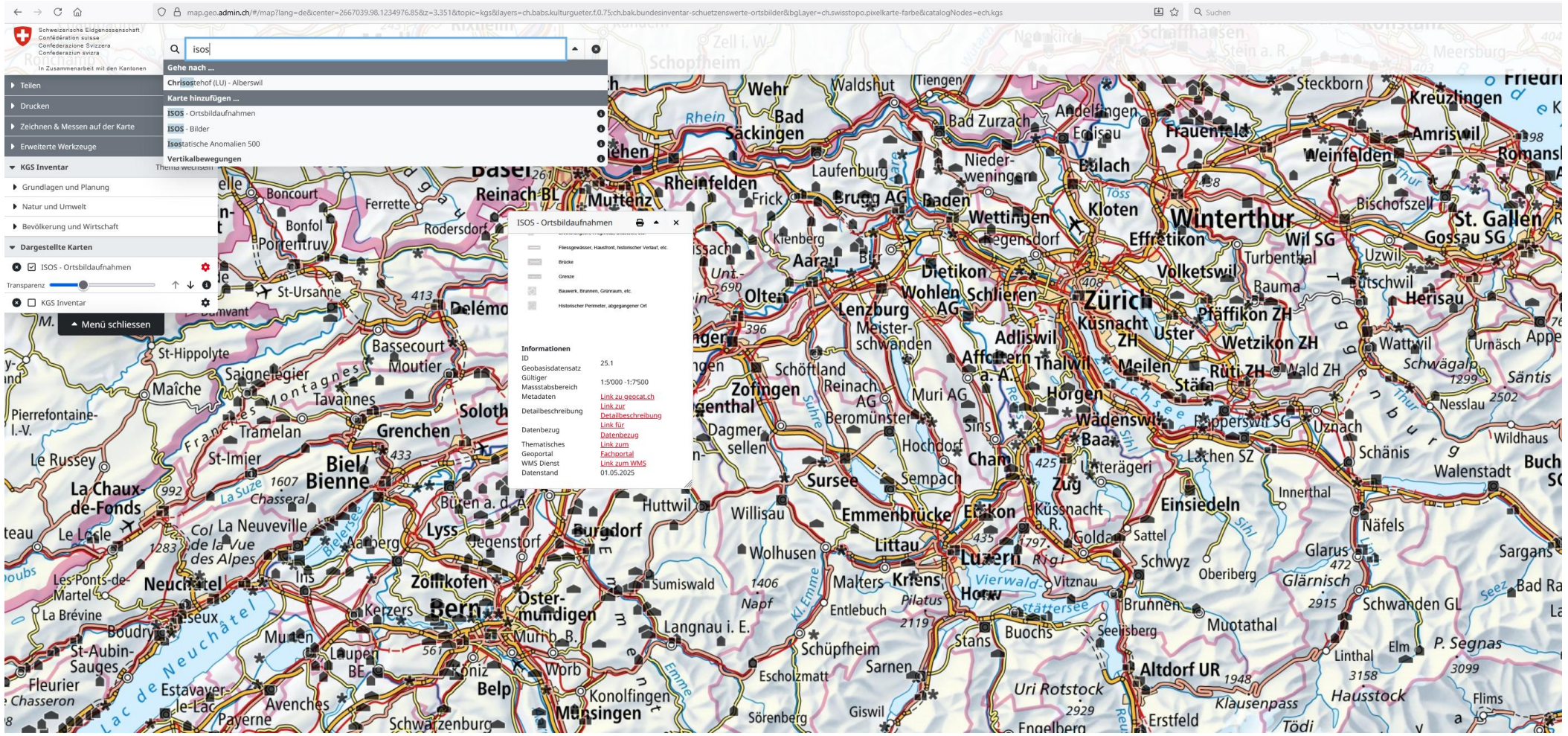
ISOS: import by ili2pg

Merge them by using SQL

Download the KGS (Kulturgüterschutz) data set from map.geo.admin.ch → besides XTF & XML, download the corresponding INTERLIS model



Download also the ISOS data set from map.geo.admin.ch. For this, we look up the model online on models.geo.admin.ch.



Please find your credentials on the Polybox

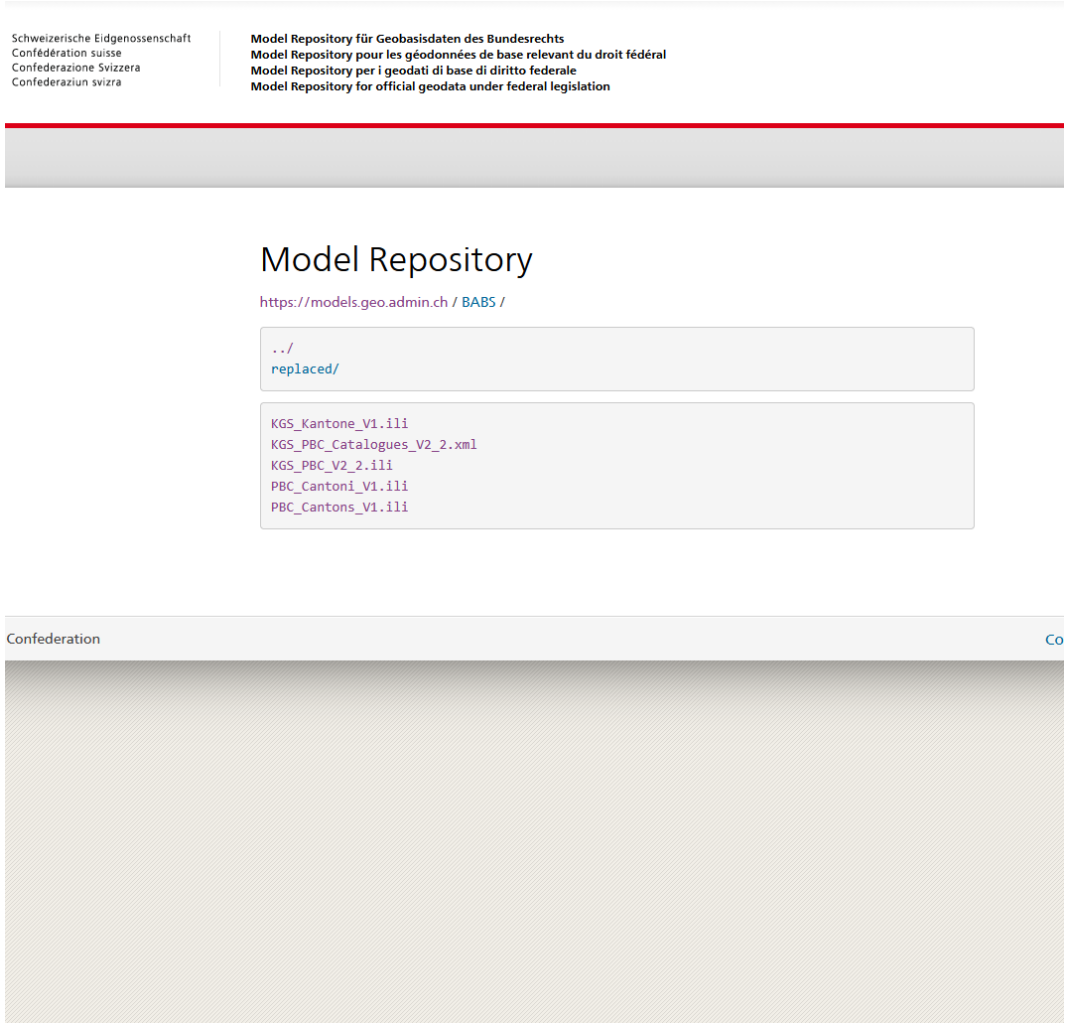
- Download the secret .env files from the Polybox service (ETH Zurich)
- You will be assigned a number from 01 to 30.
- After having noted your number, you need two files:
  - common.env (contains the general database information)
  - iat25_{01-30}.env (your personal credentials for writing onto the database)

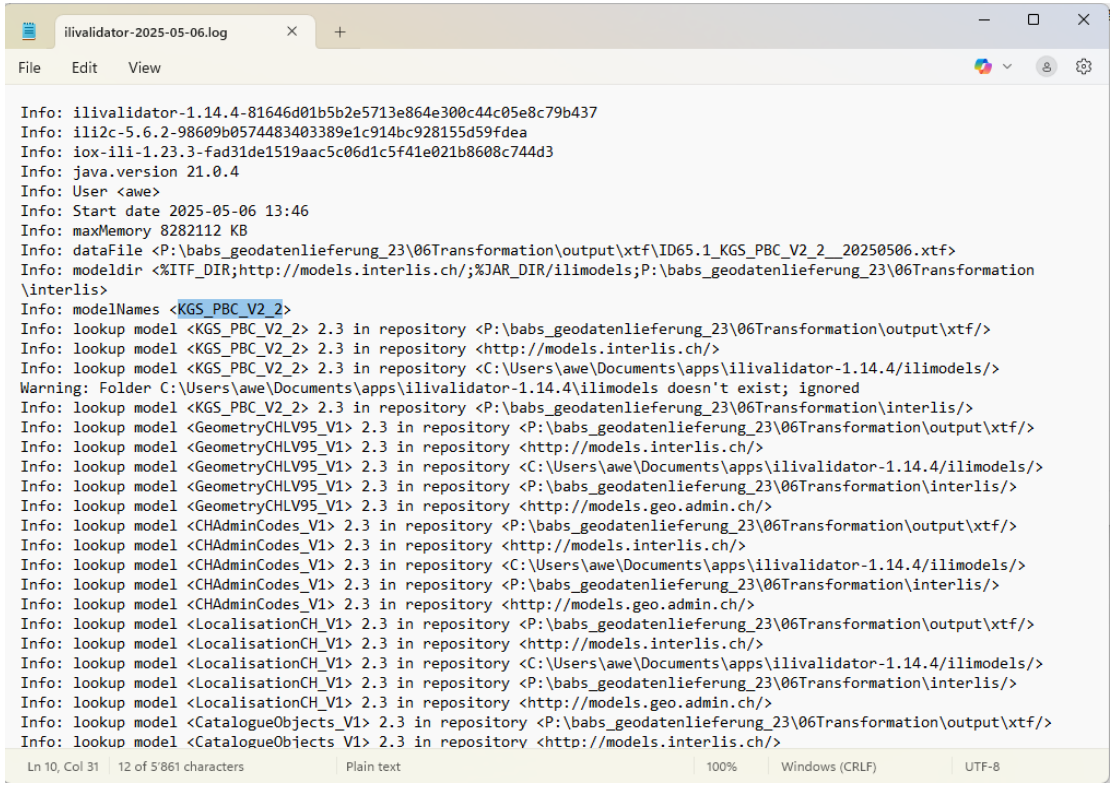
ETH zürich

Part 1: Import the KGS data set by using Model Baker

geoinformation engineering ikg eth zurich

Search and download the INTERLIS model from models.geo.admin.ch → select BABS, but which one is the correct one?





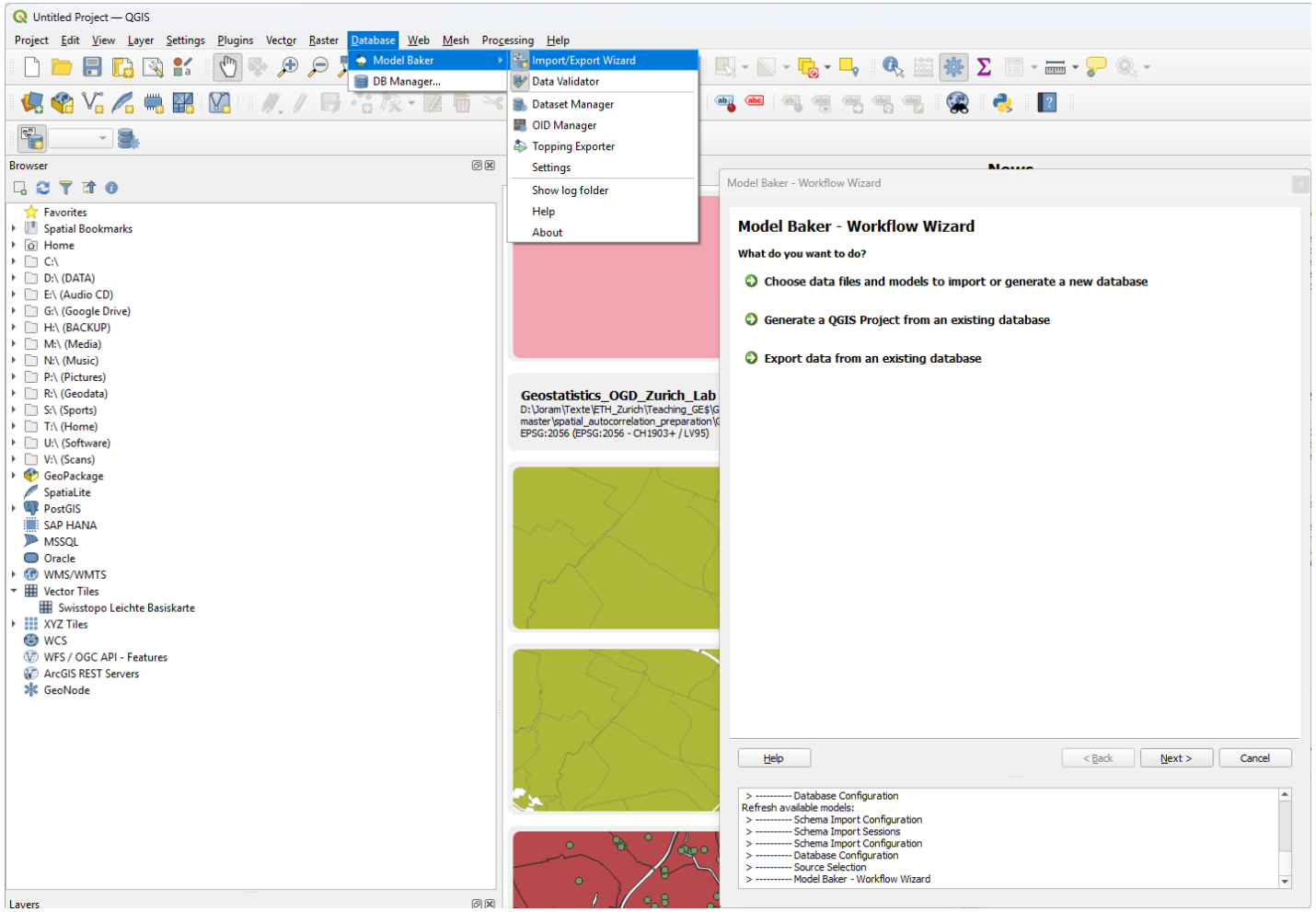
The decisive hint comes from the ilivalidator file, as it mentions the model used to build the XTF exchange file.

OK, clicked on the link, but what is that?
→ ILI file = INTERLIS model. Copy-paste and save it with the ending ILI.

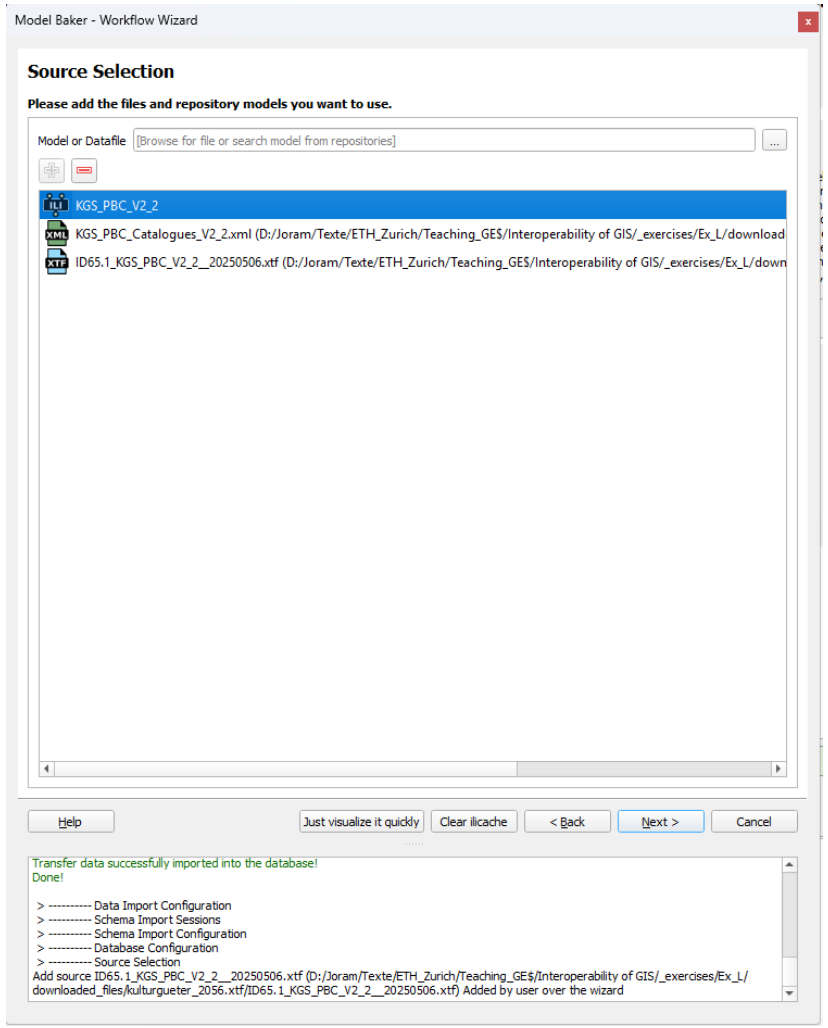
```
INTERLIS 2.3;

!! Version    | Who   | Modification
!!------------------------------------------------------------------------------
!! 2025-01-07 | KOGIS | Attribut Bemerkung auf 500 Zeichen erhöht. Attribut Beschreibung auf 1000 Zeichen erhöht

!!@ furtherInformation=https://www.babs.admin.ch/de/aufgabenbabs/kgs/inventar.html
!!@ IDGeoIV=65.1
!!@ technicalContact=mailto:Geographisches-Informationssystem@babs.admin.ch

MODEL KGS_PBC_V2_2 (de)
AT "https://models.geo.admin.ch/BABS/"
VERSION "2025-01-07"  =
  IMPORTS CatalogueObjects_V1,LocalisationCH_V1,GeometryCHLV95_V1,CHAdminCodes_V1;

  TOPIC KGS_Codelisten =

    CLASS Objektarten_Catalogue
    EXTENDS CatalogueObjects_V1.Catalogues.Item =
      /** [DE] Numerischer Code der Objektart
       * [FR] Code numérique du type d'objet
       */
      Objektcode : MANDATORY 0 .. 9999;
      /** [DE] Mehrsprachige Objektbezeichnung
       * [FR] Désignation multilingue de l'objet
       */
      Objektbezeichnung : MANDATORY LocalisationCH_V1.MultilingualText;
      UNIQUE Objektcode;
    END Objektarten_Catalogue;

    STRUCTURE Objektarten_CatRef
    EXTENDS CatalogueObjects_V1.Catalogues.MandatoryCatalogueReference =
      Reference (EXTENDED) : MANDATORY REFERENCE TO (EXTERNAL) Objektarten_Catalogue;
    END Objektarten_CatRef;

  END KGS_Codelisten;

  TOPIC KGS_Inventar =
    DEPENDS ON KGS_PBC_V2_2.KGS_Codelisten;

    STRUCTURE EGID_Structure =
      /** [DE] Eidgenössischer Gebäudeidentifikator
       * [FR] Identificateur fédéral de bâtiment
       */
      EGID : MANDATORY 1 .. 900000000;
    END EGID_Structure;

    STRUCTURE Adressen_Structure =
      /** [DE] Postadresse mit Strassenname und Hausnummer (Bezeichnung gemäss dem amtlichen Gebäudeverzeichnis)
       * [FR] Adresse postale avec nom de rue et numéro de maison (notation selon le répertoire officiel du bâtime
       */
      Adresse : MANDATORY TEXT*100;
    END Adressen_Structure;
```

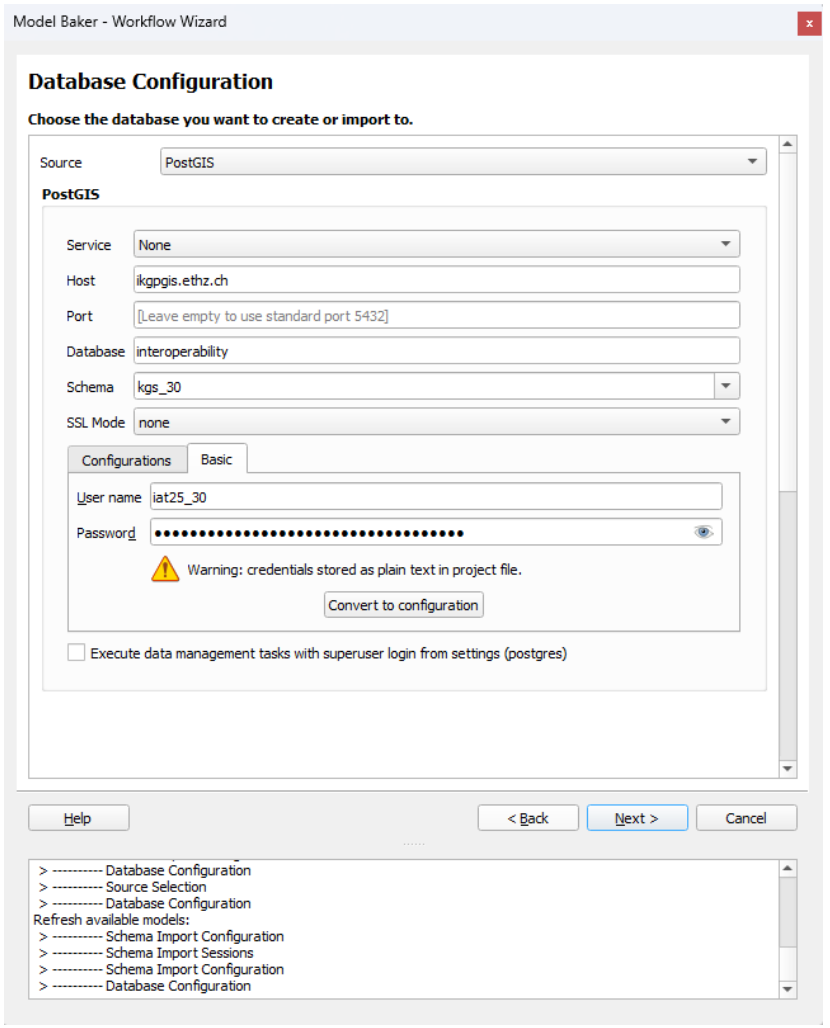
1. Open a notepad and paste the copied ILI code.
2. Save the file without TXT ending (*.*), but by choosing the ending “*.ili”.
3. Alternatively, the

In QGIS, install and load the Model Baker plugin. Then, start the Import/Export Wizard

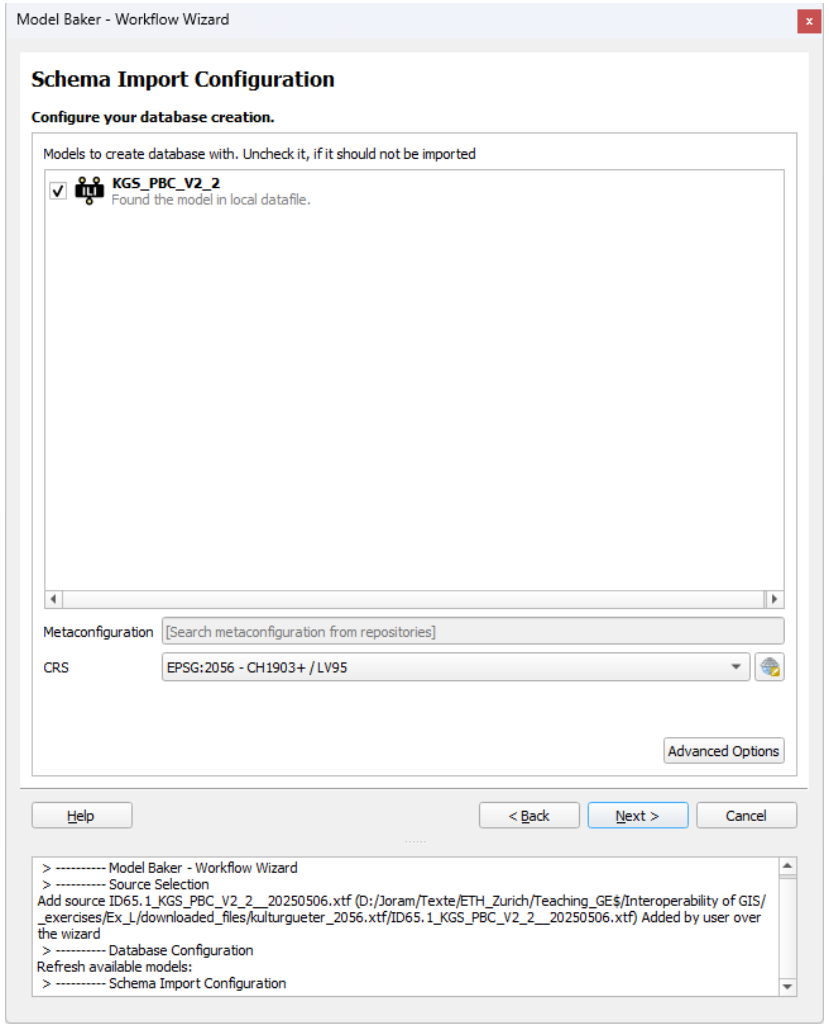


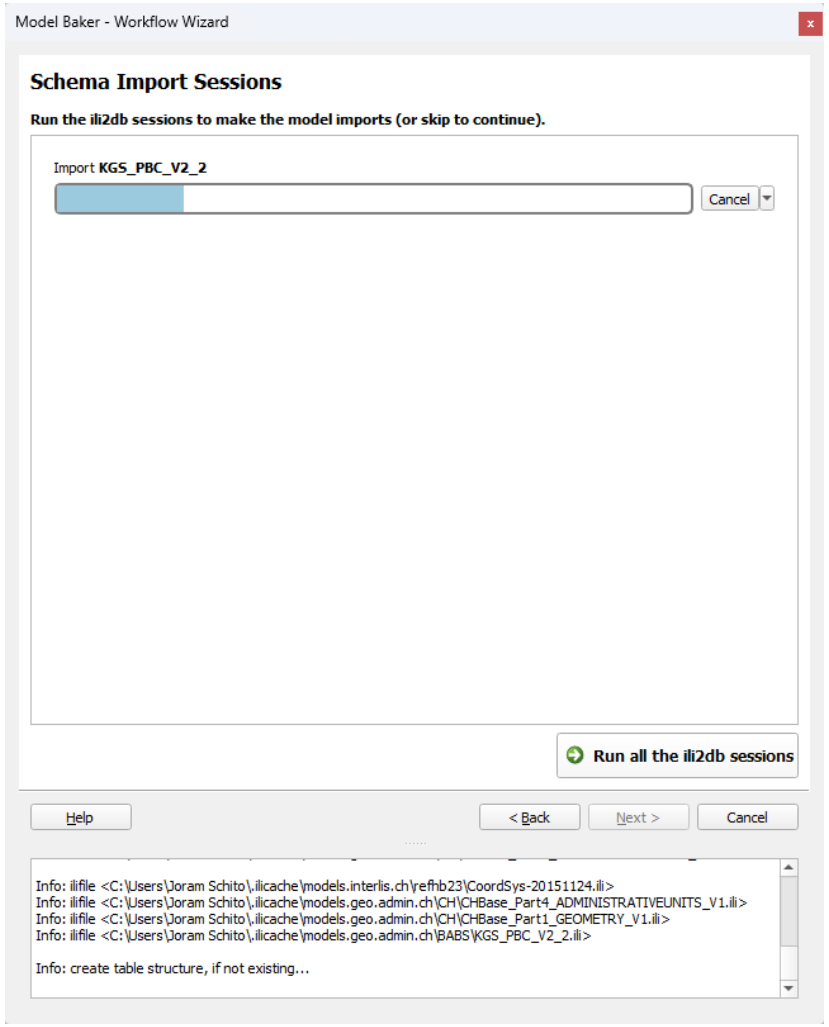
Load the downloaded XTF (data), XML (catalogue) and ILI (model) files. Then, insert your user credentials to write into the database.



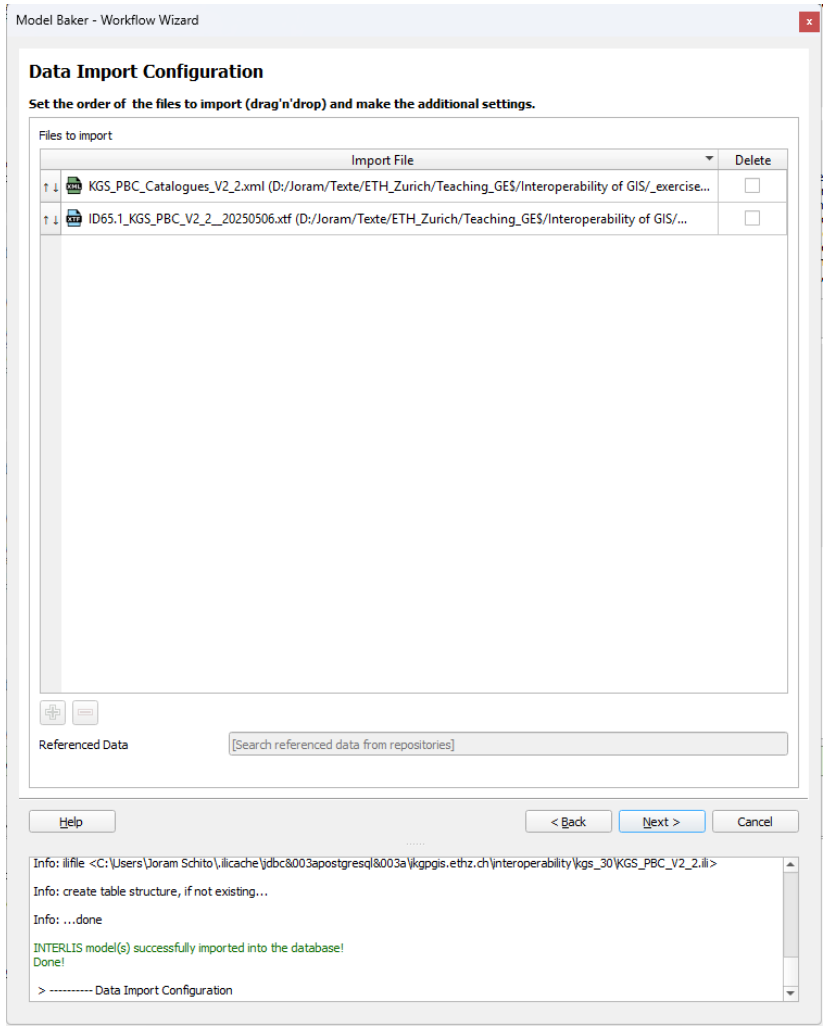


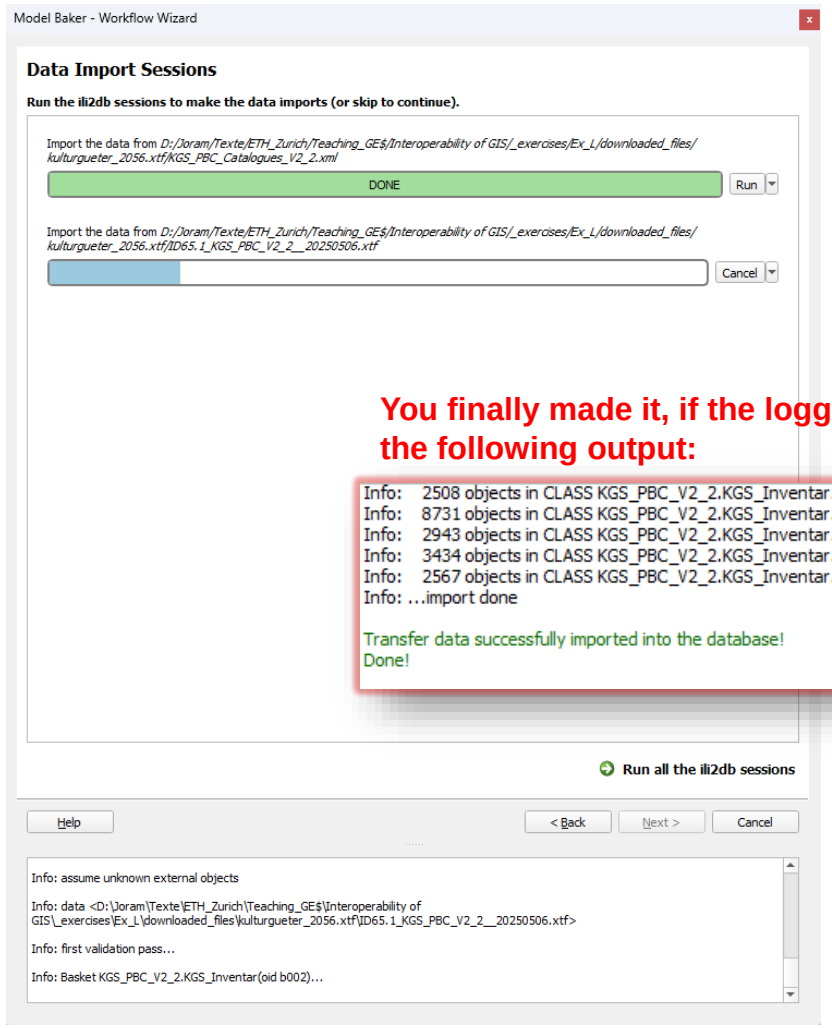
Write the empty tabular structure provided by the ILI model to the database by starting an ili2db session.





Now, import first the XML and then the XTF (or ITF) file(s) by starting an ili2db session. This process may take a while.

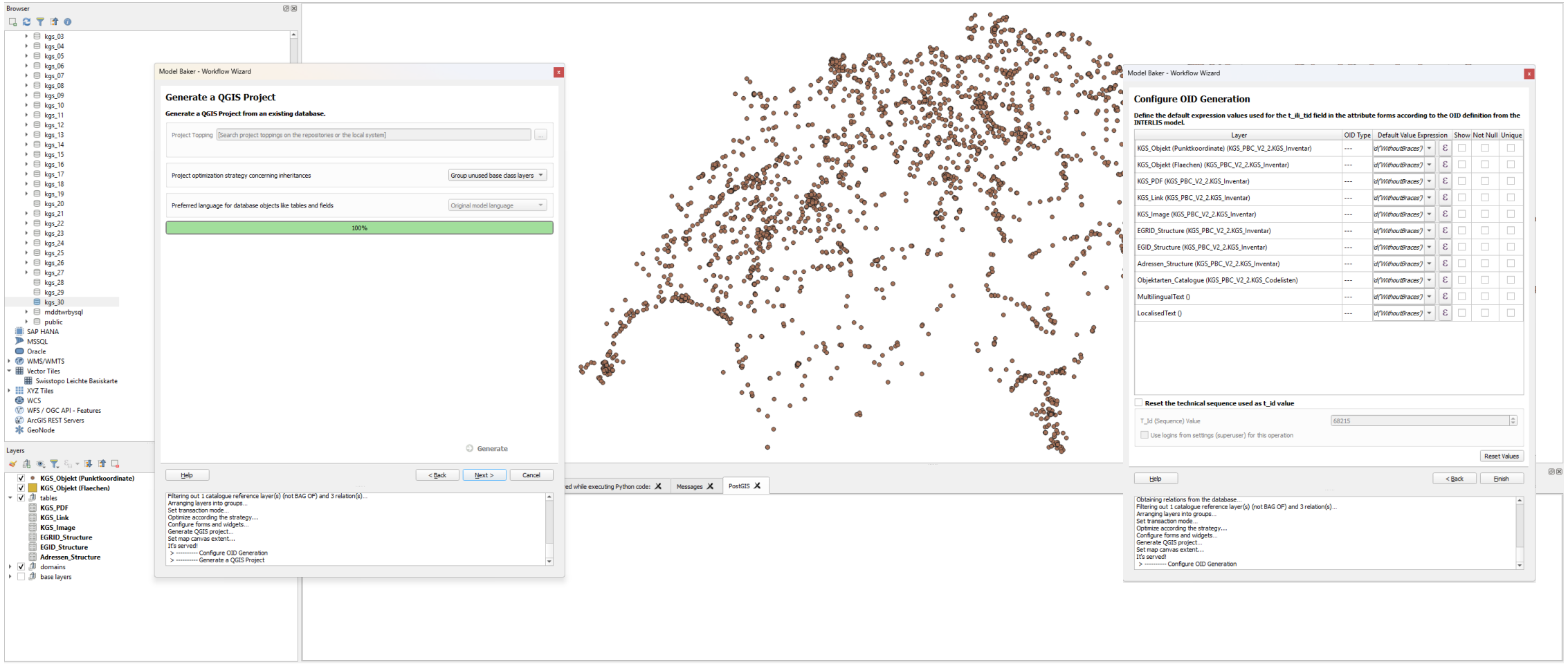




You finally made it, if the logger shows the following output:

```
Info:   2508 objects in CLASS KGS_PBC_V2_2.KGS_Inventar.EGID_Structure
Info:   8731 objects in CLASS KGS_PBC_V2_2.KGS_Inventar.KGS_Image
Info:   2943 objects in CLASS KGS_PBC_V2_2.KGS_Inventar.KGS_Link
Info:   3434 objects in CLASS KGS_PBC_V2_2.KGS_Inventar.KGS_Objekt
Info:   2567 objects in CLASS KGS_PBC_V2_2.KGS_Inventar.KGS_PDF
Info: ...import done

Transfer data successfully imported into the database!
Done!
```

Finally, choose what to do with the unused tables and make QGIS project out of the data. The data should have been correctly loaded.



Alternative by using ili2db (procedure for Windows): Copy this code, adapt the paths and the credentials, and run the code in CMD

import model

```
"C:\Program Files\Amazon Corretto\jdk17.0.16_8\bin\java" -jar
"C:\Users\User_XYZ\AppData\Roaming\QGIS\QGIS3\profiles\default\python\plugins\QgisModelBaker\libs\modelbaker\iliwrapper\bin\ili2pg-5.3.1\ili2pg-5.3.1.jar" --schemaimport --dbhost ikgpgis.ethz.ch --dbusr iat25_XX --dbpwd ***** --dbdatabase interoperability --dbschema isos_XX --coalesceCatalogueRef --createNumChecks --createUnique --createFk --createFkIdx --coalesceMultiSurface --coalesceMultiLine --coalesceMultiPoint --coalesceArray --beautifyEnumDispName --createGeomIdx --createMetaInfo --expandMultilingual --createTypeConstraint --createEnumTabsWithId --createTidCol --smart2Inheritance --strokeArcs --createBasketCol=False --defaultSrsAuth EPSG --defaultSrsCode 2056 --preScript NULL --postScript NULL --createNlsTab --models KGS_PBC_V2_2 --iliMetaAttrs NULL
```

import XML

```
"C:\Program Files\Amazon Corretto\jdk17.0.16_8\bin\java" -jar
"C:\Users\User_XYZ\AppData\Roaming\QGIS\QGIS3\profiles\default\python\plugins\QgisModelBaker\libs\modelbaker\iliwrapper\bin\ili2pg-5.3.1\ili2pg-5.3.1.jar" --import --importTid --dbhost ikgpgis.ethz.ch --dbusr iat25_XX --dbpwd *****
--dbdatabase interoperability --dbschema kgs_XX --iliMetaAttrs NULL
"C:/Path/To/Your/Unzipped/kulturgueter_2056.xtf/KGS_PBC_Catalogues_V2_2.xml"
```

import XTF

```
"C:\Program Files\Amazon Corretto\jdk17.0.16_8\bin\java" -jar
"C:\Users\User_XYZ\AppData\Roaming\QGIS\QGIS3\profiles\default\python\plugins\QgisModelBaker\libs\modelbaker\iliwrapper\bin\ili2pg-5.3.1\ili2pg-5.3.1.jar" --import --importTid --dbhost ikgpgis.ethz.ch --dbusr iat25_XX --dbpwd *****
--dbdatabase interoperability --dbschema kgs_XX --iliMetaAttrs NULL
"C:/Path/To/Your/Unzipped/kulturgueter_2056.xtf/ID65.1_KGS_PBC_V2_2__20250506.xtf"
```

KGS data set

Alternative by using ili2db (procedure for Mac/Linux): Copy this code, adapt the paths and the credentials, and run the code in CMD

import model

```
java -jar "/home/username/.local/share/QGIS/QGIS3/profiles/default/python/plugins/QgisModelBaker/libs/modelbaker/iliwrapper/bin/ili2pg-5.3.1/ili2pg-5.3.1.jar" --schemaimport --dbhost ikgpgis.ethz.ch --dbusr iat25_XX --dbpwd ***** --dbdatabase interoperability --dbschema isos_XX --coalesceCatalogueRef --createNumChecks --createUnique --createFk --createFkIdx --coalesceMultiSurface --coalesceMultiLine --coalesceMultiPoint --coalesceArray --beautifyEnumDispName --createGeomIdx --createMetaInfo --expandMultilingual --createTypeConstraint --createEnumTabsWithId --createTidCol --smart2Inheritance --strokeArcs --createBasketCol=False --defaultSrsAuth EPSG --defaultSrsCode 2056 --preScript NULL --postScript NULL --createNlsTab --models KGS_PBC_V2_2 --iliMetaAttrs NULL
```

import XML

```
java -jar "/home/username/.local/share/QGIS/QGIS3/profiles/default/python/plugins/QgisModelBaker/libs/modelbaker/iliwrapper/bin/ili2pg-5.3.1/ili2pg-5.3.1.jar" --import --importTid --dbhost ikgpgis.ethz.ch --dbusr iat25_XX --dbpwd ***** --dbdatabase interoperability --dbschema kgs_XX --iliMetaAttrs NULL "home/path/to/your/unzipped/kulturgueter_2056.xtf/KGS_PBC_Catalogues_V2_2.xml"
```

import XTF

```
java -jar "/home/username/.local/share/QGIS/QGIS3/profiles/default/python/plugins/QgisModelBaker/libs/modelbaker/iliwrapper/bin/ili2pg-5.3.1/ili2pg-5.3.1.jar" --import --importTid --dbhost ikgpgis.ethz.ch --dbusr iat25_XX --dbpwd ***** --dbdatabase interoperability --dbschema kgs_XX --iliMetaAttrs NULL "home/path/to/your/unzipped/kulturgueter_2056.xtf/ID65.1_KGS_PBC_V2_2__20250506.xtf"
```

KGS data set

ETH zürich

Part 2: Import the ISOS data set by using ili2db

geoinformation engineering ikg eth zurich

Requirements for using ili2db

ili2db

INTERLIS 2-loader for databases

ili2pg, ili2gpkg and ili2fgdb are programs that write an INTERLIS transfer file according to an INTERLIS model into a database (PostgrSQL/PostGIS, GeoPackage or ESRI FileGDB) or create such a transfer file from a database.

Software errors of ili2db or new tool requirements can be registered directly on the developer platform GitHub.

- ili2pg (for PostgreSQL/PostGIS)
- ili2gpkg (for GeoPackage)
- ili2fgdb (for ESRI FileGDB)

Download **ili2pg** – Version 5.0.1 of 25.09.2023, ZIP

Download **ili2gpkg** – Version 5.0.1 of 25.09.2023, ZIP

Download **ili2fgdb** – Version 5.0.1 of 25.09.2023, ZIP

- Java JRE 8 or Java JDK 8 (e.g., Amazon Corretto 8)
- ili2db (standalone), provided for exporting INTERLIS data to:
  - PostgreSQL (ili2pg)
  - GPKG (ili2gpkg)
  - Esri File Geodatabase (ili2fgdb)
  - FME (ili2fme, already included in FME)
- Model Baker plugin for QGIS, as it includes the ili2db package

Procedure for Windows: Copy this code, adapt the paths and the credentials, and run the code in CMD

import model

```
"C:\Program Files\Amazon Corretto\jdk17.0.16_8\bin\java" -jar
"C:\Users\User_XYZ\AppData\Roaming\QGIS\QGIS3\profiles\default\python\plugins\QgisModelBaker\libs\modelbaker\iliwrap
per\bin\ili2pg-5.3.1\ili2pg-5.3.1.jar" --schemaimport --dbhost ikgpgis.ethz.ch --dbusr iat25_XX --dbpwd ***** --
dbdatabase interoperability --dbschema isos_XX --coalesceCatalogueRef --createNumChecks --createUnique --createFk --
createFkIdx --coalesceMultiSurface --coalesceMultiLine --coalesceMultiPoint --coalesceArray --beautifyEnumDispName -
-createGeomIdx --createMetaInfo --expandMultilingual --createTypeConstraint --createEnumTabsWithId --createTidCol --
smart2Inheritance --strokeArcs --createBasketCol=False --defaultSrsAuth EPSG --defaultSrsCode 2056 --preScript NULL
--postScript NULL --createNlsTab --models ISOS_V2 --iliMetaAttrs NULL
```

import XML

```
"C:\Program Files\Amazon Corretto\jdk17.0.16_8\bin\java" -jar "C:\Users\ User_XYZ
\AppData\Roaming\QGIS\QGIS3\profiles\default\python\plugins\QgisModelBaker\libs\modelbaker\iliwrapper\bin\ili2pg-
5.3.1\ili2pg-5.3.1.jar" --import --importTid --dbhost ikgpgis.ethz.ch --dbusr iat25_XX --dbpwd ***** --dbdatabase
interoperability --dbschema isos_XX --iliMetaAttrs NULL "C:/Path/To/Your/Unzipped/bundesinventar-schuetzenswerte-
ortsbilder_2056.xtf/ISOS_Catalogues_V2_20220426.xml"
```

import XTF

```
"C:\Program Files\Amazon Corretto\jdk17.0.16_8\bin\java" -jar "C:\Users\ User_XYZ
\AppData\Roaming\QGIS\QGIS3\profiles\default\python\plugins\QgisModelBaker\libs\modelbaker\iliwrapper\bin\ili2pg-
5.3.1\ili2pg-5.3.1.jar" --import --importTid --dbhost ikgpgis.ethz.ch --dbusr iat25_XX --dbpwd ***** --dbdatabase
interoperability --dbschema isos_XX --iliMetaAttrs NULL "C:/Path/To/Your/Unzipped/bundesinventar-schuetzenswerte-
ortsbilder_2056.xtf/ISOS_V2_20250304.xtf"
```

ISOS data set

For those who cannot use neither QGIS + Model Baker nor ili2db on their device, we set up a “magic hand” that does everything for you…

- Go to this temporary webpage and run the command.
- Remark: The service does basically the same as the script `import_data_to_schema.bat`, which you can run directly in your console by using the following syntax (for number 10):
  - `import_data_to_schema.bat 10 kgs`
  - `import_data_to_schema.bat 10 isos`

ETH zürich

Part 3: Merge both data sets by defined mapping rules via SQL

geoinformation engineering ikg eth zurich

Aufgabe 1 – KGS-Objekte mit Schutzwürdigkeit A extrahieren

- **Lernziel:** Enums via FK-Tabelle korrekt auflösen; Selektionsabfrage; Ergebnistabelle erzeugen.
- **Gegeben:**
  - `kgs_28.kgs_objekt` (Enthält `kgs_kategorie` als FK)
  - `kgs_28.kgs_objekt_kgs_kategorie` mit `t_id`, `ilicode` (`A.A`, `A.verstaerkter_Schutz`, `B`)
- **Ziel:** Erzeuge `result_28.kgs_objekte_a` mit den Spalten:
  - `objekt_nr`, `kurztext`, `kanton`, `gemeinde`, `punktkoordinate`
- **Hinweis:** Kategorie `A` umfasst `A.A` **und** `A.verstaerkter_Schutz`.

```
-- Loesungsvorschlag
CREATE TABLE
result_28.kgs_objekte_a ASSELECT
o.objekt_nr, o.kurztext, o.kanton,
o.gemeinde, o.punktkoordinateFROM
kgs_28.kgs_objekt oJOIN
kgs_28.kgs_objekt_kgs_kategorie e
ON o.kgs_kategorie = e.t_idWHERE
e.ilicode LIKE 'A.%';
```

```
-- Check
SELECT ilicode, COUNT(*) FROM
kgs_28.kgs_objekt o JOIN
kgs_28.kgs_objekt_kgs_kategorie e
ON o.kgs_kategorie=e.t_idWHERE
e.ilicode LIKE 'A.%'GROUP BY
ilicode;
```

Aufgabe 2 – Anzahl KGS-A pro Gemeinde (Aggregation)

- **Lernziel:** Gruppierung, Aggregation, Ergebnistabelle mit Ranking.
- **Gegeben:** Ergebnis aus Aufgabe 1 oder Originaltabellen.
- **Ziel:** Erzeuge `result_28.kgs_count_a_pro_gemeinde`:
  - `gemeinde`, `anzahl_kgs_a` (absteigend sortiert)

```
-- Loesungsvorschlag
CREATE TABLE
result_28.kgs_count_a_pro_gemeinde
ASSELECT  o.gemeinde,  COUNT(*) AS
anzahl_kgs_aFROM kgs_28.kgs_objekt
oJOIN
kgs_28.kgs_objekt_kgs_kategorie e
ON o.kgs_kategorie = e.t_idWHERE
e.ilicode LIKE 'A.%'GROUP BY
o.gemeindeORDER BY anzahl_kgs_a
DESC;
```

```
-- Check
CREATE TABLE
result_28.kgs_count_a_pro_gemeinde
ASSELECT gemeinde, COUNT(*) AS
anzahl_kgs_aFROM
result_28.kgs_objekte_aGROUP BY
gemeindeORDER BY anzahl_kgs_a DESC;
```

Aufgabe 3 – Unterkategorien von A je Kanton

- **Lernziel:** Abgeleitete Attribute bilden, Feinklassifikation auswerten.
- **Gegeben:** Tabellen wie in Aufgabe 1.
- **Ziel:** Erzeuge `result_28.kgs_a_sub_by_kanton`:
  - `kanton`, `kategorie_sub`, `anzahl`, wobei `kategorie_sub` ∈ {`A.A`, `A.verstaerkter_Schutz`}

```
-- Loesungsvorschlag
CREATE TABLE
result_28.kgs_a_sub_by_kanton
ASSELECT  o.kanton,  e.ilicode AS
kategorie_sub,  COUNT(*)  AS
anzahlFROM kgs_28.kgs_objekt oJOIN
kgs_28.kgs_objekt_kgs_kategorie e
ON o.kgs_kategorie = e.t_idWHERE
e.ilicode LIKE 'A.%'GROUP BY
o.kanton, e.ilicodeORDER BY
o.kanton, e.ilicode;
```

```
-- Check
SELECT kanton,  SUM(CASE WHEN
kategorie_sub='A.A' THEN anzahl
ELSE 0 END) AS a_a,  SUM(CASE WHEN
kategorie_sub='A.verstaerkter_Schut
z' THEN anzahl ELSE 0 END) AS
a_verstFROM
result_28.kgs_a_sub_by_kantonGROUP
BY kantonORDER BY kanton;
```

Aufgabe 4 – Objektarten anreichern (deutsche Bezeichnungen) und je Objekt aggregieren

- **Lernziel:** BAG-Relationen auflösen, Katalogreferenzen joinen, Sprachfilter auf MultilingualText, Stringaggregation.
- **Gegeben:**
  - `kgs_28.kgs_objekt`
  - `objektarten_catref` mit Spalten `kgs_objekt_objektart` (Owner-FK auf `kgs_objekt.t_id`) und `areference` (FK auf Katalog)
  - `objektarten_catalogue` mit `t_id`, `objektcode` (numerisch)
- **Ziel:** Erzeuge `result_28.kgs_objekt_mit_objektarten` mit Spalten:
  - `objekt_nr`, `gemeinde`, `kanton`, `punktkoordinate`, `objektarten_codes`, wobei `objektarten_codes` eine kommaseparierte Liste der Katalogcodes (z. B. 111, 718, …) pro Objekt ist

```
CREATE TABLE
result_28.kgs_objekt_mit_objektarte
n ASSELECT  o.objekt_nr,
o.gemeinde,  o.kanton,
o.punktkoordinate,  STRING_AGG(
DISTINCT c.objektcode::text,    ',
' ORDER BY c.objektcode::text  ) AS
objektarten_codesFROM
kgs_28.kgs_objekt AS oJOIN
kgs_28.objektarten_catref AS r  ON
r.kgs_objekt_objektart = o.t_idJOIN
kgs_28.objektarten_catalogue AS c
ON c.t_id = r.areferenceGROUP BY
o.objekt_nr, o.gemeinde, o.kanton,
o.punktkoordinate;
```

Aufgabe 5 – Räumliche Integration: KGS-A innerhalb von ISOS-Ortsbild-Perimetern

- **Lernziel:** Räumlicher Join (Punkt-in-Polygon), BAG-Perimeter von ISOS zusammen mit KGS nutzen.
- **Gegeben:**
  - `isos_28.geometrie_perimeter` mit Geometriespalte `perimeter` und Owner-FK `isos_v2isos_ortsbild_geometrie_perimeter` (Zeiger auf das zugehörige Ortsbild; numerischer Schlüssel)
  - `kgs_28.kgs_objekt` und Enum-Tabelle wie in Aufgabe 1
- **Ziel:** Erzeuge `result_28.isos_ortsbilder_mit_kgs_count` mit:
  - `ob_tid` (Owner-Schluessel aus `geometrie_perimeter.isos_v2isos_ortsbild_geometrie_perimeter`), `kgs_in_ob` (Anzahl KGS-Objekte mit Kategorie A innerhalb des jeweiligen Ortsbild-Perimeters)

```
CREATE TABLE result_28.isos_obperimeter_fk_mit_kgs_count AS
WITH gp_norm AS (
  SELECT
    gp.isos_v2isos_ortsbild_geometrie_perimeter AS ob_tid,
    -- SRID normalisieren (du hast 2056, falls 0 → setze 2056)
    CASE
      WHEN ST_SRID(gp.perimeter)=0 THEN ST_SetSRID(gp.perimeter,2056)
      WHEN ST_SRID(gp.perimeter)<>2056 THEN ST_Transform(gp.perimeter,2056)
      ELSE gp.perimeter
    END AS per_2056
  FROM isos_28.geometrie_perimeter gp
  WHERE gp.isos_v2isos_ortsbild_geometrie_perimeter IS NOT NULL
),
ob_poly AS (
  SELECT
    ob_tid,
    ST_UnaryUnion(
      ST_Collect(
        ST_CollectionExtract(
          ST_MakeValid(ST_CurveToLine(per_2056))
        , 3)
      )
    ) AS geom
  FROM gp_norm
  GROUP BY ob_tid
),
kgs_pts AS (
  SELECT
    k.objekt_nr,
    CASE
      WHEN ST_SRID(k.punktkoordinate)=0 THEN ST_SetSRID(k.punktkoordinate,2056)
      WHEN ST_SRID(k.punktkoordinate)<>2056 THEN
ST_Transform(k.punktkoordinate,2056)
      ELSE k.punktkoordinate
    END AS geom
  FROM kgs_28.kgs_objekt k
  JOIN kgs_28.kgs_objekt_kgs_kategorie e
    ON e.t_id = k.kgs_kategorie
  WHERE e.ilicode LIKE 'A.%'
    AND k.punktkoordinate IS NOT NULL
)
SELECT
  p.ob_tid,                    -- das ist der "Ortsbild-Schluessel" aus
geometrie_perimeter
  COUNT(k.objekt_nr) AS kgs_in_ob
FROM ob_poly p
LEFT JOIN kgs_pts k
  ON ST_Covers(p.geom, k.geom)  -- Rand inklusive (robuster als Within)
GROUP BY p.ob_tid
ORDER BY kgs_in_ob DESC;
```

Aufgabe 6 – Vereinheitlichung der Punktobjekte von KGS und ISOS

- **Lernziel:** Daten aus verschiedenen INTERLIS-Modellen vereinheitlichen, Beziehungen über Fremdschlüssel herstellen, ENUM-Referenzen (CHCantonCode) auflösen
- **Gegeben:**
  - `isos_28.ortsbild` und `isos_28.isos_v2isos_ortsbild`
    - Punktgeometrie: koordinaten
    - Gemeinde: aname
    - Zuordnung zu isos_28.kanton, das über acode mit isos_28.chcantoncode.t_id verknüpft ist
    - Attribut dispname enthält den sichtbaren Kantonscode (BE, ZH, GR, ...)
  - `kgs_28.kgs_objekt`
    - Punktgeometrie: punktkoordinate
    - Kanton als Referenz auf kgs_28.chcantoncode
    - Gemeinde als Textfeld gemeinde

```
CREATE TABLE result_28.kulturerbe_punkte AS
/* -------- KGS: Kanton via dispname aus kgs_28.chcantoncode -------- */
WITH kgs_pts AS (
  SELECT
    'KGS_Objekt'::text      AS src,
    o.objekt_nr::text       AS src_id,
    kc.dispname::text       AS kanton,
    o.gemeinde::text        AS gemeinde,
    (
      CASE
        WHEN ST_SRID(o.punktkoordinate)=0     THEN ST_SetSRID(o.punktkoordinate,2056)
        WHEN ST_SRID(o.punktkoordinate)<>2056 THEN ST_Transform(o.punktkoordinate,2056)
        ELSE o.punktkoordinate
      END
    )::geometry(Point,2056)  AS geom
  FROM kgs_28.kgs_objekt o
  LEFT JOIN kgs_28.chcantoncode kc
    ON kc.t_id = o.kanton
  WHERE o.punktkoordinate IS NOT NULL
),

/* -------- ISOS: Kanton pro Ortsbild (Aggregation ueber evtl. mehrere Kantone) -------- */
isos_ob_kanton AS (
  SELECT
    ob.t_id                                          AS ob_tid,
    STRING_AGG(DISTINCT cc.dispname::text, ', '
               ORDER BY cc.dispname::text)           AS kanton_disp
  FROM isos_28.ortsbild ob
  JOIN isos_28.kanton k
    ON k.ortsbild_kantone = ob.t_id
  JOIN isos_28.chcantoncode cc
    ON cc.t_id = k.acode   -- deine Beobachtung: acode == chcantoncode.t_id
  GROUP BY ob.t_id
),
/* -------- ISOS: analog fuer die zweite Punktquelle -------- */
isos_ob_v2_kanton AS (
  SELECT
    ob2.t_id                                         AS ob2_tid,
    STRING_AGG(DISTINCT cc.dispname::text, ', '
               ORDER BY cc.dispname::text)           AS kanton_disp
  FROM isos_28.isos_v2isos_ortsbild ob2
  JOIN isos_28.kanton k
    ON k.isos_v2isos_ortsbild_kantone = ob2.t_id
  JOIN isos_28.chcantoncode cc
    ON cc.t_id = k.acode
  GROUP BY ob2.t_id
),

/* -------- ISOS: Punkte mit Kanton (dispname) und Gemeinde (aname) -------- */
isos_ob_pts AS (
  SELECT
    'ISOS_Ortsbild'::text      AS src,
    ob.id::text                AS src_id,
    COALESCE(k.kanton_disp,'') AS kanton,          -- falls kein Eintrag, leer
    ob.aname::text             AS gemeinde,        -- wie von dir vorgeschlagen
    (
      CASE
        WHEN ST_SRID(ob.koordinaten)=0     THEN ST_SetSRID(ob.koordinaten,2056)
        WHEN ST_SRID(ob.koordinaten)<>2056 THEN ST_Transform(ob.koordinaten,2056)
        ELSE ob.koordinaten
      END
    )::geometry(Point,2056)    AS geom
  FROM isos_28.ortsbild ob
  LEFT JOIN isos_ob_kanton k
    ON k.ob_tid = ob.t_id
  WHERE ob.koordinaten IS NOT NULL
),

/* -------- ISOS: Punkte mit Kanton (dispname) und Gemeinde (aname) -------- */
isos_ob_v2_pts AS (
  SELECT
    'ISOS_Ortsbild_v2'::text   AS src,
    ob2.id::text               AS src_id,
    COALESCE(k.kanton_disp,'') AS kanton,
    ob2.aname::text            AS gemeinde,
    (
      CASE
        WHEN ST_SRID(ob2.koordinaten)=0     THEN ST_SetSRID(ob2.koordinaten,2056)
        WHEN ST_SRID(ob2.koordinaten)<>2056 THEN ST_Transform(ob2.koordinaten,2056)
        ELSE ob2.koordinaten
      END
    )::geometry(Point,2056)    AS geom
  FROM isos_28.isos_v2isos_ortsbild ob2
  LEFT JOIN isos_ob_v2_kanton k
    ON k.ob2_tid = ob2.t_id
  WHERE ob2.koordinaten IS NOT NULL
),

/* -------- Merge aller drei Quellen -------- */
merged AS (
  SELECT src, src_id, kanton, gemeinde, geom FROM kgs_pts
  UNION ALL
  SELECT src, src_id, kanton, gemeinde, geom FROM isos_ob_pts
  UNION ALL
  SELECT src, src_id, kanton, gemeinde, geom FROM isos_ob_v2_pts
)
SELECT DISTINCT ON (src, src_id)
  src, src_id, NULLIF(kanton,'') AS kanton,  -- leere Strings wieder auf NULL
  gemeinde, geom
FROM merged;

-- sinnvolle Indices
CREATE INDEX IF NOT EXISTS ix_kulturerbe_punkte_v2_geom   ON result_28.kulturerbe_punkte_v2 USING GIST (geom);
CREATE INDEX IF NOT EXISTS ix_kulturerbe_punkte_v2_src_id ON result_28.kulturerbe_punkte_v2 (src, src_id);
```

Aufgabe 6 – Vereinheitlichung der Punktobjekte von KGS und ISOS

- **Ziel:** Erstelle eine vereinheitlichte Ergebnistabelle: `result_28.kulturerbe_punkte` mit den Spalten:
  - src: Herkunftsquelle ('KGS_Objekt', 'ISOS_Ortsbild', 'ISOS_Ortsbild_v2')
  - src_id: Identifikator (objekt_nr bzw. id)
  - kanton: Kanton (Kürzel aus dispname)
  - gemeinde: Gemeindename (gemeinde aus KGS bzw. aname aus ISOS)
  - geom: Geometriepunkt (LV95 / SRID 2056)

```
CREATE TABLE result_28.kulturerbe_punkte AS
/* -------- KGS: Kanton via dispname aus kgs_28.chcantoncode -------- */
WITH kgs_pts AS (
  SELECT
    'KGS_Objekt'::text      AS src,
    o.objekt_nr::text       AS src_id,
    kc.dispname::text       AS kanton,
    o.gemeinde::text        AS gemeinde,
    (
      CASE
        WHEN ST_SRID(o.punktkoordinate)=0     THEN ST_SetSRID(o.punktkoordinate,2056)
        WHEN ST_SRID(o.punktkoordinate)<>2056 THEN ST_Transform(o.punktkoordinate,2056)
        ELSE o.punktkoordinate
      END
    )::geometry(Point,2056)  AS geom
  FROM kgs_28.kgs_objekt o
  LEFT JOIN kgs_28.chcantoncode kc
    ON kc.t_id = o.kanton
  WHERE o.punktkoordinate IS NOT NULL
),

/* -------- ISOS: Kanton pro Ortsbild (Aggregation ueber evtl. mehrere Kantone) -------- */
isos_ob_kanton AS (
  SELECT
    ob.t_id                                         AS ob_tid,
    STRING_AGG(DISTINCT cc.dispname::text, ', '
               ORDER BY cc.dispname::text)          AS kanton_disp
  FROM isos_28.ortsbild ob
  JOIN isos_28.kanton k
    ON k.ortsbild_kantone = ob.t_id
  JOIN isos_28.chcantoncode cc
    ON cc.t_id = k.acode   -- deine Beobachtung: acode == chcantoncode.t_id
  GROUP BY ob.t_id
),
/* -------- ISOS: analog fuer die zweite Punktquelle -------- */
isos_ob_v2_kanton AS (
  SELECT
    ob2.t_id                                        AS ob2_tid,
    STRING_AGG(DISTINCT cc.dispname::text, ', '
               ORDER BY cc.dispname::text)          AS kanton_disp
  FROM isos_28.isos_v2isos_ortsbild ob2
  JOIN isos_28.kanton k
    ON k.isos_v2isos_ortsbild_kantone = ob2.t_id
  JOIN isos_28.chcantoncode cc
    ON cc.t_id = k.acode
  GROUP BY ob2.t_id
),

/* -------- ISOS: Punkte mit Kanton (dispname) und Gemeinde (aname) -------- */
isos_ob_pts AS (
  SELECT
    'ISOS_Ortsbild'::text      AS src,
    ob.id::text                AS src_id,
    COALESCE(k.kanton_disp,'') AS kanton,          -- falls kein Eintrag, leer
    ob.aname::text             AS gemeinde,        -- wie von dir vorgeschlagen
    (
      CASE
        WHEN ST_SRID(ob.koordinaten)=0     THEN ST_SetSRID(ob.koordinaten,2056)
        WHEN ST_SRID(ob.koordinaten)<>2056 THEN ST_Transform(ob.koordinaten,2056)
        ELSE ob.koordinaten
      END
    )::geometry(Point,2056)    AS geom
  FROM isos_28.ortsbild ob
  LEFT JOIN isos_ob_kanton k
    ON k.ob_tid = ob.t_id
  WHERE ob.koordinaten IS NOT NULL
),

/* -------- ISOS: Punkte mit Kanton (dispname) und Gemeinde (aname) -------- */
isos_ob_v2_pts AS (
  SELECT
    'ISOS_Ortsbild_v2'::text   AS src,
    ob2.id::text               AS src_id,
    COALESCE(k.kanton_disp,'') AS kanton,
    ob2.aname::text            AS gemeinde,
    (
      CASE
        WHEN ST_SRID(ob2.koordinaten)=0     THEN ST_SetSRID(ob2.koordinaten,2056)
        WHEN ST_SRID(ob2.koordinaten)<>2056 THEN ST_Transform(ob2.koordinaten,2056)
        ELSE ob2.koordinaten
      END
    )::geometry(Point,2056)    AS geom
  FROM isos_28.isos_v2isos_ortsbild ob2
  LEFT JOIN isos_ob_v2_kanton k
    ON k.ob2_tid = ob2.t_id
  WHERE ob2.koordinaten IS NOT NULL
),

/* -------- Merge aller drei Quellen -------- */
merged AS (
  SELECT src, src_id, kanton, gemeinde, geom FROM kgs_pts
  UNION ALL
  SELECT src, src_id, kanton, gemeinde, geom FROM isos_ob_pts
  UNION ALL
  SELECT src, src_id, kanton, gemeinde, geom FROM isos_ob_v2_pts
)
SELECT DISTINCT ON (src, src_id)
  src, src_id, NULLIF(kanton,'') AS kanton,  -- leere Strings wieder auf NULL
  gemeinde, geom
FROM merged;

-- sinnvolle Indices
CREATE INDEX IF NOT EXISTS ix_kulturerbe_punkte_v2_geom   ON result_28.kulturerbe_punkte_v2 USING GIST (geom);
CREATE INDEX IF NOT EXISTS ix_kulturerbe_punkte_v2_src_id ON result_28.kulturerbe_punkte_v2 (src, src_id);
```

Dr. Joram Schito
Lecturer | Spatial Data Scientist
jschito@ethz.ch

ETH Zurich
Institute of Cartography and Geoinformation
HIL G 23.1
Stefano-Franscini-Platz 5
8093 Zurich, Switzerland

www.gis.ethz.ch