TECHNISCHE UNIVERSITÄT WIEN

MASTERARBEIT

# Archiving Digital Maps with GeoPackage and Vector-tile Dissemination

Ausgeführt am Department für
Geodäsie und Geoinformation
der Technischen Universität Wien

unter der Anleitung von
**Univ.Prof. Mag.rer.nat. Dr.rer.nat. Georg Gartner, TU Wien**
und
**Dipl.-Ing. Dr. Markus Jobst, Bundesamt für Eich- und Vermessungswesen**
**Dr.-Ing. Christian Murphy, TU München**

durch

**Yunnan Chen**
Schulwinkel 4, Stuttgart

28.03.2019

Unterschrift (Student)


MASTER'S THESIS

# Archiving Digital Maps with GeoPackage and Vector-tile Dissemination

Conducted at the Department of
Geodesy and Geoinformation
Technical University Vienna

Under the supervision of
**Univ.Prof. Mag.rer.nat. Dr.rer.nat. Georg Gartner, TU Vienna**
and
**Dipl.-Ing. Dr. Markus Jobst, Federal Office of Metrology and Surveying**
**Dr.-Ing. Christian Murphy, TU Munich**

by

**Yunnan Chen**
Schulwinkel 4, Stuttgart

28.03.2019

Signature (Student)

# Statement of Authorship

Herewith I declare that I am the sole author of the submitted Master's thesis entitled:

"**Archiving Digital Maps with GeoPackage and Vector-tile Dissemination**"

I have fully referenced the ideas and work of others, whether published or unpublished. Literal or analogous citations are clearly marked as such.

Vienna, 28.03.2019 Yunnan Chen

# Acknowledgements

The last two years in the International Cartography M.Sc. programme have been such a special journey. This master's thesis could not be accomplished without the supports of many people.

First, I would like to express my sincere gratitude and appreciation to my first supervisor and the deputy head of Information Management Department at Austrian Federal Office for Metrology and Surveying, Dr. Markus Jobst, who has been providing marvelous guidance, ideas, support, and suggestions in the last few months.

I would also like to show my sincere appreciation to the Chair of the Research Group of Cartography at TU Wien, Prof. Georg Gartner. Thank you for all your incredible help and support along the way.

Special thanks go to Dr.-Ing. Christian Murphy, Dr. Corné van Elzakker, and M.Sc. Juliane Cron, M.Sc. Francisco Porras for your kind help, support, and suggestions during the thesis assessment meetings.

Last, I would like to thank my family and friends for their unconditional support during this master's program.

# Abstract

The preservation of digital maps has been a long-term challenge for the cartographic community, GIS industry, and related mapping agencies. As a special type of digital data, digital maps not only contains geospatial datasets, but also requires the metadata and styling information associated with datasets. However, research has shown that most digital mapping archive projects in the world are using the shapefile and GeoTIFF format to store geospatial data, which cannot completely fulfill the principles of digital map preservation. Thus, an alternative method is necessary to satisfy the need for digital map preservation.

This thesis proposes a new approach to archive digital maps using GeoPackage format and to disseminate digital maps using vector tile format. For this thesis, a case study was conducted using the Austrian Cartographic Model with national map styling. The new approach allows the archiving and sharing processes to be conducted in a free, cross-platform, and open-source environment using QGIS and Geoserver. In addition, a Python script has been developed to provide a user-friendly interface and to minimize the human effort during the process of embedding metadata.

Based on the results from the case study, this thesis proposes an innovative approach to embed relevant metadata for future usage in digital maps using GeoPackage and to embed styling from GeoPackage when distributing digital maps that use vector-tile format. It also shows the possibility of using Python scripts to automate the procedure of embedding relevant information into GeoPackage.



Keywords: Digital Map Preservation, Geopackage, Metadata, Map Style, Vector Tile, Python

# Table of Contents

# List of Figures

# List of Tables

# Abbreviations

| | |
|---|---|
| API | Application programming interface |
| BEV | National Mapping Agency of Austria |
| CSS | Cascading Style Sheets |
| CSV | Comma-separated Values File |
| DEM | Digital Elevation Model |
| ESRI | Environmental Systems Research Institute |
| FGDC | Federal Geographic Data Committee (USA) |
| FTP | File Transfer Protocol |
| GIS | Geographic Information System |
| GML | Geography Markup Language |
| GUI | Graphical User Interface |
| INSPIRE | Infrastructure for Spatial Information in Europe |
| MSDIS | Missouri Spatial Data Information Service |
| NDIPPP | National Digital Information Infrastructure and Preservation Program |
| NSDI | National Spatial Data Infrastructure |
| NGDA | National Geospatial Digital Archives (USA) |
| OGC | Open Geospatial Consortium |
| PDF | Portable Document Format |
| PNG | Portable Network Graphics |
| SDI | Spatial Data Infrastructure |
| SHP | ESRI Shapefile Format |
| SLD | Styled Layer Descriptor |
| SOA | Service-Oriented Architecture |
| SQL | Structured Query Language |
| SVG | Scalable Vector Language |
| TIFF | Tagged Image File Format |
| TXT | Standard Text Document |
| UTF | Unicode Transformation Format |
| VML | Vector Markup Language |
| WFS | Web Feature Service |
| WMS | Web Map Service |
| XML | Extensible Markup Language |
| ZIP | Archive File Format |

# 1. Introduction

## 1.1 Background

*"The long-term preservation of digital records has been the topic of study by librarians and archivists for the last twenty years or so (Brand, 2000; The Commission, 1996)."*

*"One special subset of digital data of particular concern is digital geospatial data."*

*"Geospatial data are unique in the digital world because real-world phenomena are stored as points, lines and polygons; and relationships between these entities are stored as part of the electronic data structure."*

*"Currently, there are no easy answers as how best to guarantee that digital data will last into the future without active management to preserve the data."* (Bleakly, D. R., 2002)

Denise R. Bleakly wrote the sentences above in her report back in 2002, which illustrated the issues of the long-term preservation of digital maps at that time. As time passes by and technology advances, cartographers are facing new challenges for the same topic. This dissertation studies the principle and status quo of archiving digital map nowadays, discusses the current main geospatial data formats, and illustrates a more promising data format for archiving and sharing digital maps in the future.

## 1.2 Motivation and Problem Statement

The preservation of digital maps has been a long-term challenge for the cartographic community, GIS industry, and relating mapping agency ever since the invention of personal computers in the 1970s. The following years, until the 21th century, has been referred as the dark ages of archiving digital cartographic heritage by Lauriault et al., (2011). In that time period, there was a major shift from paper maps to digital maps, which had a strong influence on information management. The issues for digital preservation involve media, technological obsolescence, data refreshing, data migration, emulation, data storage versus data access, and long-term costs (Bleakly, 2002). Due to the technical limitations such as the availability of personal computer and internet, the digital maps and geospatial data were limited to professionals at that time. Meanwhile, the public was mainly using the paper map provided by mapping agencies. As time passes by, the situation and challenges of archiving digital maps have changed as well.

Currently, as a result of the development of the Internet and the increasing use of computers and mobile devices, there is a major transition, and the use of GIS services are shifting from a desktop environment to the web service side, which allows more public users to engage in the mapping activities. In addition, as the size of spatial data increases, the data management becomes

more complicated and has changed from local database management to distributed data management (Ružicka, 2016). Thus, data format, as the fundamental part of mapping service, has inevitable influences on the future digital mapping and distributed GIS. It is essential to study the geospatial data formats that have been widely used in digital map preservation nowadays.

Based on multiple reports, digital map archive projects are usually carried out by government agencies and most of the archive projects use shapefile format (SHP) and GeoTIFF to store digital data. In the last decade, there has been an increasing number of governments joining open government partnership and legislating Sunshine Acts for geographic data. According to Bernard (2013), the City of Vienna became the first government agency among all German speaking countries to start an open government initiative and to accept the concepts of open government data, which allowed public to access the government data for personal use. Later, the Austrian National Mapping Agency (BEV) also joined the act by sharing its geodata in shapefile format and GeoTIFF format and hosting the web service of the Austrian Cartographic Model with national map styling. However, apart from viewing the map at different scales and offering a general measurement tool, the BEV web map service does not allow too much user interaction such as viewing the feature attributes. Also, although the spatial data provided by BEV does include the metadata as PDF in the folder, it lacks style information and certain layers.

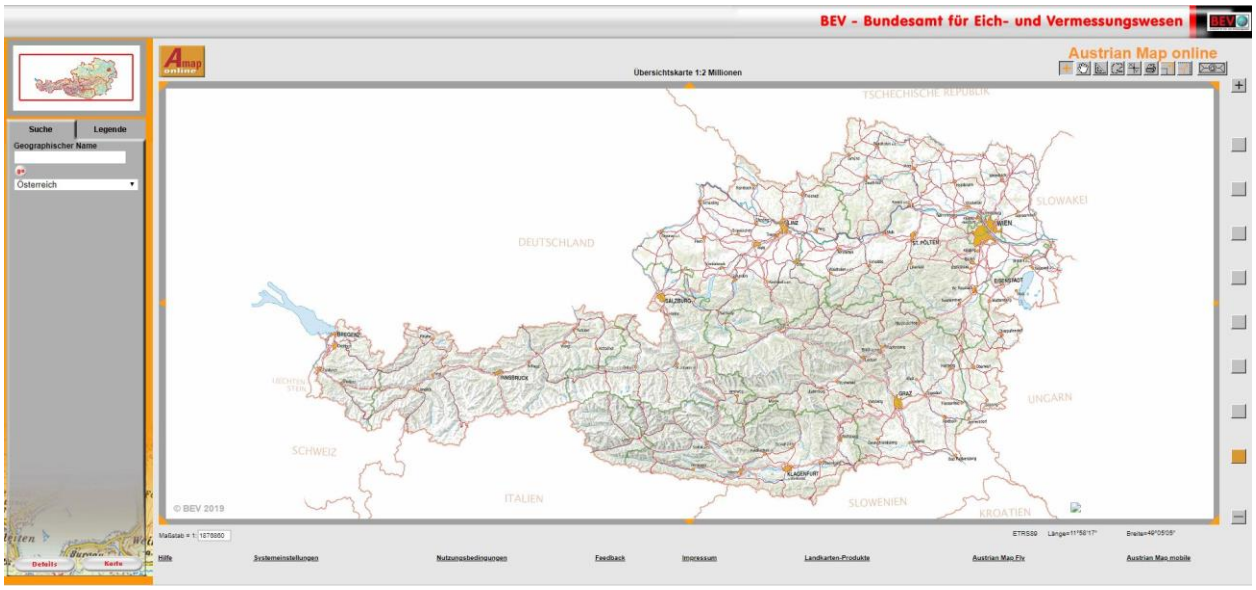



Figure 1-1 Austria Map Online (AMap Online) – A Web Map Viewer hosted by BEV
http://www.austrianmap.at/amap/index.php

However, ever since ESRI introduced the SHP in the beginning of 1990s, shapefile has gradually become the most popular file format for storing geographic vector data. According to ARC Advisory Group, ESRI is, without doubt, the largest GIS Company in the industry by taking 43 percent of the market share, while the second-largest supplier counts only an 11 percent share (Alban, 2015). As the data format invented by ESRI, shapefile has primary advantages, which are

fast drawing capability and readily available supportability (Theobald, 2001). Despite the advantages, the SHP does have many drawbacks that make it unsuitable for the future digital map archiving and distribution. Cepicky et al.,(2017) listed several fatal disadvantages of the shapefile format such as no coordinate reference system by default, multifile format, 4GB size limit, non-topological format, and poor support for attribute data types. Thus, it is necessary to find an alternative data format to replace the role of SHP in archiving digital maps.

In addition to vector data, the raster data format is equally important in digital map preservation and dissemination. Many progressive transmission techniques for raster data have been introduced. The efficiency of those techniques has made it possible to build the most of web map services using raster data, which allows user to access spatial information freely (Antoniou et al., 2009). With all the advantages that raster data had, it seemed that raster tiles held the future for web mapping. However, the traditional raster maps, whether it is driven from vector images or cache tiles for aerial images, lack the ability and function to interact with users through web application. For example, users could not retrieve information from map object. Besides, it was also impossible for uses to style the map in their own way. Therefore, the current raster data format has also posed potential threats to future digital map preservation and dissemination.

In sum, as Veenendaal et al.,(2017) stated in their research, the challenge for future geospatial data archiving and distribution is to create an integrated, interactive, and automated environment that makes it easier for user to access information and knowledge. Thus, the main challenge for this master thesis is to find a suitable data format, platform, and method, which would be able to archive, publish, and distribute digital maps without the needs to put huge effort to embedding relevant information or to reestablish styling like is needed with original shapefile format.

## 1.3 Research Questions and Objective

Based on the motivation and problem statement above, the ultimate goal of this thesis is to explore an improved method for long-term archiving and distribution of digital maps. The thesis will use the cartographic model and national map style from BEV as a case study. Therefore, the current archiving format used by BEV will be examined and a suitable spatial data format for future archiving will be determined that can integrate original spatial data in SHP provided by BEV with relevant information such as metadata and style. Programming scripts will be created to minimize the human effort during the procedure.

To be more specific, the research objectives will be divided into three parts below.

1. *To determine and analyze a suitable data format that has the ability to store geospatial data and its relevant information such as metadata and style.*
   According to Rashidun (2015), GeoPackage(*.gpkg), a new GIS data format, was designed to provide widespread support and the use of a single spatial data file format

by both commercial and non-commercial platforms. Unlike other current formats, this data format is non-platforms-specific, thus increasing the interoperability and options for geospatial sharing.

2. *To establish a suitable method for archiving, publishing, and distributing digital maps with customized styling.*

3. *To determine a suitable programming and to develop a script that minimizes the human effort during the procedure.*
   Python is an interpreted, cross-platform, and open-source programming language, which is also known as the glue language to integrate and develop extension components for other programming languages (Sanner, 1998)

In order to achieve the objectives, the following research questions will be answered:

- Is it possible to embed relevant metadata for future usage in digital maps using GeoPackage?
- Is it possible to embed styling from GeoPackage when distributing digital maps that use vector-tile format?
- Is it possible to create a Python script that partially automate the procedures of embedding relevant information into GeoPackage?

## 1.4 Overview of Contents

This thesis contains six chapters. The following chapter explores the principles and challenges of archiving digital maps by examining former studies. In addition, it evaluates the most common data format for archiving digital maps and proposes an alternative data format for future digital map preservation. The third chapter explains the adopted methodology through a case study. Chapter four presents the result of the case study and experiment. The fifth chapter states the problems that happened during the experiment and discusses the potential work for the future. The last chapter summarizes the main points of evidences.

# 2. Literature Review and Theoretical Background

## 2.1 Principle of Digital Maps Preservation and Archiving

Digital map archiving or preservation is the act of storing and maintaining the integrity of geospatial datasets, its metadata, and styling for future use. Preservation activity usually involves government policies or regulations, which helps to set up archiving standard to ensure the integrity and long-term accessibility of digital maps.

Even though digital maps are indeed a type of digital data, geospatial data has special characteristics, so a different approach is required to preserve them, compared to regular data preservation methods. Janée(2009) stated that geospatial data has several characteristics relating to its preservation: 1. There is no uniform data model that geospatial data could be vector and raster, discrete and continuous, topological and non-topological; 2. Geospatial data have proprietary formats which means they are strongly associated with the applications; 3. Geospatial data have multiple granule sizes that ranges from individual features to thematic layers of features; 4. Geospatial data are usually stored in databases which are relational system with geographic extensions; 5. The size of geospatial datasets are growing along with technological development; 6. Geospatial datasets may last for a long time period, especially for long-term programs such as Landsat Program; 7. Geospatial data requires extensive context for it to be interpreted in the future; 8. Geospatial data may contain implicit context; 9. Geospatial data can be dynamic, which may periodically require reprocessing.

Bleakly also explained the uniqueness of geospatial data. In his research, Bleakly (2002) wrote, *"geospatial data store in a GIS usually have relationships between objects stored as part of the data structure", "Geospatial data are multi-scaled and have multi-resolutions", "Geospatial data can be both current and historical, and the large amounts of geospatial data that could be preserved and archived could prove to be very valuable to future researchers looking for long-term changes in the environment or ecosystems", "geospatial data can be in multiple formats".*

Considering the special characteristics of geospatial data discussed above, Clark (2016) listed five principles for the preservation of geospatial data from an information producer's point of view (See Figure 2-1). First, because many information producers archive data retroactively, it is necessary to create metadata while the data is being created. Second, a comprehensive geospatial metadata scheme is required so that future users can fully understand the context of the geospatial data. Third, completeness and simplicity need to be considered for data management planning. Fourth, archiving a cartographic representation of geospatial data not only provides context for future users but it also helps the future producer to understand which parts need to be archived

from a large information package. Fifth, it is important to preserve the geospatial dataset in its current data format but also allow it to be reproduced in diverse data structures in the future.

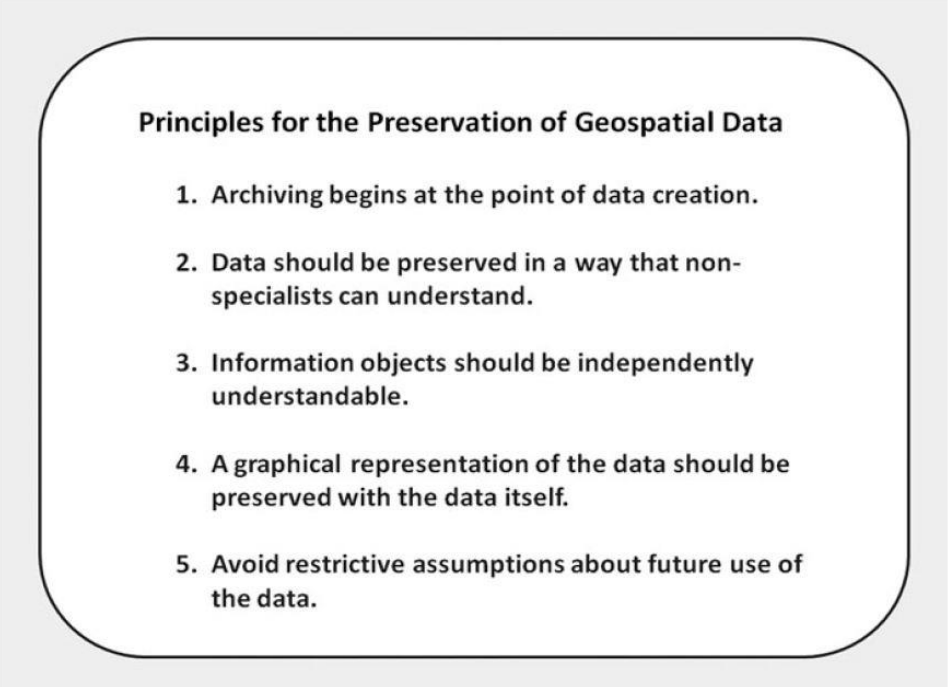



Figure 2-1. Five principles for the preservation of geospatial data (Clark, 2016)

In addition to Clark's study, Shaon et al.,(2011) also outlines eight principles which were approved by European national mapping agencies who present their geospatial data through INSPIRE. Their principles of archiving digital maps were from the view of the lifecycle of data which are "*creation to maintenance, archival, preservation to accessing archived data*". The eight principles set by Shaon et al., are listed below:

1. *"Archiving of digital geographic information begins at the point of data creation, rather than at the point of withdrawal from active systems."*
2. *"Establishment and agreement of a common preservation planning process and s set of common preservation objectives between data producers and archives is the backbone for any archiving business case."*
3. *"Be selective and decide what to archive and what to lose."*
4. *"Consider archiving timeframes of 1, 10, 1000 years."*


Figure 2-2. Geo-archiving Lifecyle(Shaon,2011)

5. "*The output of the planning process should also be preserved over the long-term to accommodate future preservation requirements.*"
6. "*Archiving is not backup.*"
7. "*Geographical data should be preserved in a way that non geo-specialists can handle it.*"
8. "*Ensure effective management and quality assurance of the metadata associated with your data.*"

Sander (2011) also pointed out that it is more difficult to archive digital geographic information than regular digital documents. He listed five essential aims:

1. *"Digital contents of the geographical information system should be delivered completely (thematic data, geographical position) in order to insure that no information is lost."*
2. *"Digital data should be saved in simply-structured formats in order to minimize the number of future data migrations."*
3. *"Digital geographical information should be saved in common data formats that can be used without proprietary software in order to ensure access independently of the life-span of a particular company or product."*
4. *"The functionalities and features of the geographical information system should be retained in order to allow for diverse queries and retrievals."*
5. *"Historical geo data should be stored in a way that will be easily accessible by future users."*

Based on the studies above, it is apparent that the integrity of original data, metadata and cartographic representation plays a significant role in the process of archiving digital maps. The principles of archiving digital maps mentioned above shall be used as a guideline when conducting the research experiment later. The next subchapter reviews the situation and challenge of archiving digital maps.

## 2.2 Status Quo and Challenge of Archiving Digital Maps

### 2.2.1 Historical Situation and Challenge of Archiving Digital Maps

Learning from the past is always essential so that people can avoid the same mistakes. Thus, this subchapter will first examine the historical situation and challenge of archiving digital maps.

According to Cartwright (2011), before the digital age, archiving cartographic artifacts heavily involved storing paper maps. Accessing paper maps and records was similar to other library management methods. In the last quarter of the 20th century, the dark ages of archiving cartographic heritage, there was a major shift from paper maps to digital maps, which had a strong influence on information management (Bleakly, 2002 & Lauriault et al., 2011). During that period, only partial descriptions of the digital maps existed in some cases, while the original data might have vanished already in other cases. In a few cases, the original data was not lost completely, but

the cost to recover it was extremely high or inestimable. In Bleakly's study (2002), he discussed the challenges of archiving digital data from the "dark ages" from seven points of views. The first challenge was to decide which media source should be used to store digital information. Compared to paper media that can last for more than a half century, digital media back then had a shorter expected life. Meanwhile, the three basic types of digital media, which were computer hard disk, CD/DVD-ROM, and tape, had relatively small storage and cost much more than today's digital media. Second, the speed of technological advancement threatened to digital spatial data because the storage media or format might not last long enough before they were replaced by new technology. In addition, the software evolved quickly, so the compatibility of data formats and software versions became an issue, meaning earlier tools and scripts might be useless in short time. Third, there was always a risk of losing data during the data refreshing process, while transferring the data from an older version to a newer one. Fourth, data migration, the tasks of transferring digital data from one system setting to another, could also cause problems in displaying, retrieving, and reusing the data. Fifth, data archiving rather than data preservation may cause a digital dataset to lose its integrity. Sixth, true digital archives should be stored and accessed for the long term. Seventh, the cost of archiving digital maps was hard to determine as it was expected to last for a long time and the cost of labor, software, and hardware would change as the time goes by. Besides all the challenges described above, Bleakly also addressed the significance of geospatial metadata for the long-term preservation of digital maps, based on the NSDI fact sheet and FGDC metadata standards.

Bleakly described the situation and challenge of archiving digital information at that time from a macro perspective. Zaslavsky studied the subject at the same time period from a detailed technical view. Zaslavsky (2001) focused on the research issues of spatial metadata and encoding standards. For the challenge of spatial metadata, Zaslavsky illustrated the importance of retaining the data quality of metadata as possible to reduce uncertainty. In addition, when dealing with multi-scale and multi-resolution datasets, a standard specification was required for dataset management. Moreover, for multi-hierarchical collections, Zaslavsky created a dataset structure in XML format (See Figure 2-3). The structure should allow spatial querying across the whole dataset through metadata. For the challenge of encoding spatial digital data standards, despite of the lack of such encoding standards at that time, Zaslavsky stated that "*standards are needed for converting spatial data preservation formats into open-format Web presentations*." He addressed the interest in developing "*a standard XML-based protocol for spatial data encoding*". An example would be the GML 2.0, which "*makes use of the XML Schema definition language to express schema constraints, and of XLink attributes to denote relationships between spatial features*." In addition, geospatial data were usually stored in registered system-specific formats and visualized in dependent GIS software, which means the style information of digital maps relied on the GIS software. To solve such an issue, Zaslavsky explored two XML-based languages, VML and SVG, which can encode and present vector graphics. He concluded that the SVG format was more suitable for visualizing XML-encoded geospatial data, which was also platform-independent.

```
<meta_catalog>
        <hierarchy>
                <hierarchy_type>grid</hierarchy_type>
                <dataset>
                        <name>DOQ</name>
                        <hierarchy_level>1-degree cell</hierarchy_level>
                        <id></id>
                        <coordinates>
                                <SEcorner></SEcorner>
                                <SWcorner></SWcorner>
                                <NEcorner></NEcorner>
                                <NWcorner></NWcorner>
                        </coordinates>
                        <reference_to_file_location/>
                        <reference_to_file_metadata/>
                        <parent>id_of_grid_parent</parent>
                        <otherparent>
                                <hierarchy_type>administrative</hierarchy_type>
                                <id>id of parent from another hierarchy</id>
                        </otherparent>
                </dataset>
        </hierarchy>
</meta_catalog>
```

Figure 2-3. Example of Spatial Index (Zaslavsky, 2001)

The project of recovering Canada Land Inventory (CLI) was brought up as example by several researchers above to illustrate the issues and importance of archiving digital maps in the early times. The Canada Geographic Information System (CGIS), the first GIS system in the world, was designed to map and store the geospatial data for CLI by Dr. Roger Thomlinson, the "father of GIS". According to Bleakly (2002), CGIS had stored around 3500 maps by the mid-1970s. The map collections was archived in 2965 nine-track tapes. Later in 1990s, the decision was made to extract the data from the nine-track tapes and convert to the latest GIS software platform at that moment while many tapes were falling apart after long-time preservation. Although the project of recovering CLI was considered a success, "*it was very costly, very time consuming, and very technically challenging, and there was indeed some loss of data*."

### 2.2.2 Modern Situation and Challenge of Archiving Digital Maps

If the first major shift of archiving digital maps was from paper map to digital map, then the second main trend, which is still in process, is shifting from a local desktop environment to web mapping services. The rapid development of Internet, computer, and mobile technologies has allowed the public to access, edit, and publish maps on the Internet by simply using their computer or mobile device. In today's world, geospatial data and digital maps are no longer the propriety products of government agencies or GIS professionals but are available to the regular citizen.

Unlike the dark ages of archiving digital maps, many long-term digital maps preservation projects have been carried out worldwide by government agencies or universities through SDI. According to Jobst and Gartner (2011), technological development has shaped modern cartography through digital approaches, which improves the traditional approach while requiring a more complicated framework to ensure the processes of reproduction, sharing, and publishing.

Based on the statement above, a SOA-based SDI is crucial for archiving digital maps in modern times. One example of archiving digital maps is the National Geospatial Digital Archive (NGDA) Project in USA. The NGDA project was a collaborative project by University of California Santa Barbara (UCSB) and Stanford University. Both universities had come up with their own technical solutions (Figure 2-4 & 2-5) specifically focusing on the needs of preserving geospatial datasets in the U.S. In general, the NGDA project was a successful and cutting edge project that had many positive influences from technical and legal perspectives, which helped to build the foundation of modern geospatial data preservation. (Erwin & Sweetkind-Singer, 2009)

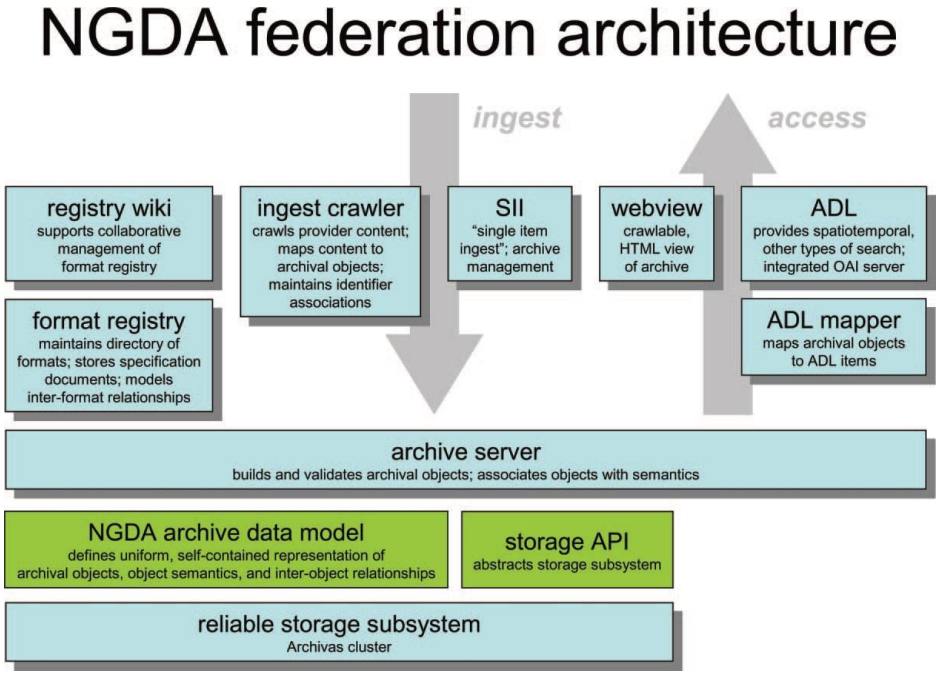



Figure 2-4. The NGDA architecture at the UCSB (Erwin & Sweetkind-Singer, 2009)

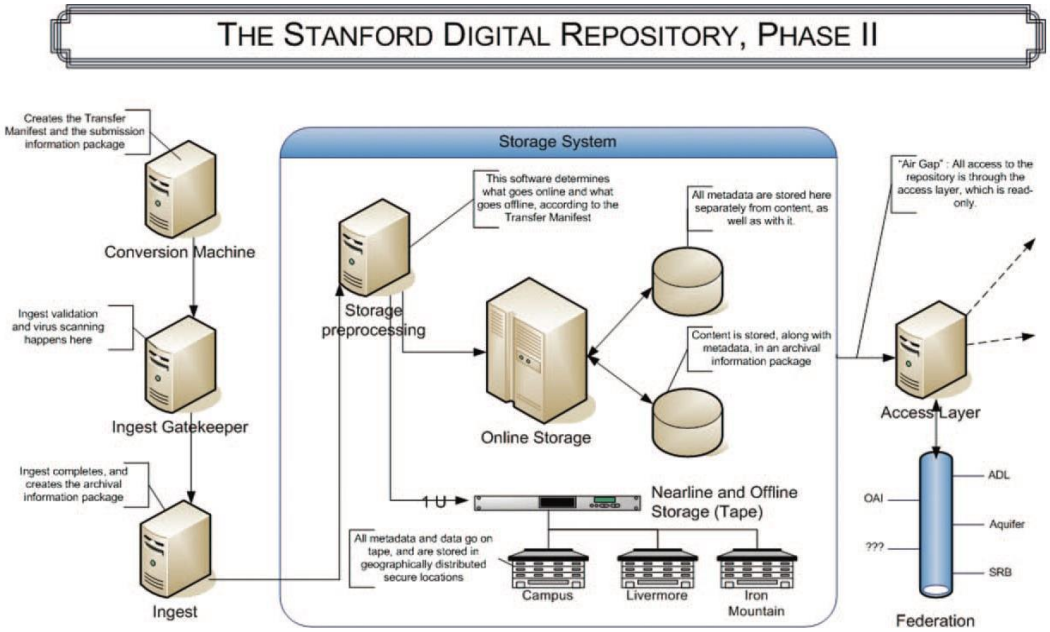



Figure 2-5. The NGDA architecture at the Stanford University (Erwin & Sweetkind-Singer, 2009)

Another example of modern digital map preservation is the Project Ellipse by Swiss Federal Archives (SFA) and the Federal Office of Topography(swisstopo). The Project Ellipse aims to fulfill the requirements of archiving geodata for different levels of government agencies. "*The goals of the project were to develop an integrated solution for all official federal geodata, to improve the long-term availability and archiving of deferral geodata, and to allow geoinformation to be restored and interpreted from archived geodata at a later date*." The digital preservation approach of the project was based on the Open Archival Information System (OAIS) model and its geospatial metadata followed the Standard SN 612050, A Swiss Metadata Model (Federal Office of Topography swisstopo, 2016).

According to Lauriault et al.,(2011), a map is one type of data representation thus the original data used to render the map is essential for cartography. Archiving geospatial data is vital for unforeseen uses in the future. Nevertheless, related technology advanced and the volume of geospatial data grew so fast that simple backup and storage of digital spatial data is no longer sufficient.

Clark (2016) stated one of the challenges of modern digital maps preservation is the proprietary nature of geospatial data formats, which raises a technical risk for long-term archiving. In addition, most geographic information relies on the context from associated datasets. The context such as metadata and styling information are typically stored in other data formats or systems. Thus, archiving individual data layer usually causes serious data loss.

In Shaon et al.,'s research, they illustrated the challenge of archiving digital maps from the view of spatial data infrastructure. Shaon et al.,(2011) wrote, "*In the context SDI, such as INSPIRE, state-of-the art service-oriented infrastructures adopt exchange formats (i.e. application schemas) that reflect domain specific conceptual data models ('feature types') rather than directly reflecting underlying database storage schemas.*" Thus, one of the challenge is to preserve the application schemas and its relationships with the relating geospatial datasets, which ensures the future accessibility of those datasets.

Locher and Termens (2012) explained three spatial data preservation challenges, which are decentralized production, complexity of data, and managing versions of the same data. First, archival projects usually involve several partners but each of them may have different interests and approaches when handling geospatial datasets. Second, the complexity of data involves "*the difficulty for human understanding and technical challenges for computer processing*". Spatial data would losses its value if humans cannot interpret it. Thus, archiving digital maps relies on metadata to provide context for future use and to sustain machine interpretability. Nevertheless, according to NGDA experience, creating metadata is usually the most time-consuming part of an archiving project. Third, thematic maps reply on reference data or a reference map to express its meaning. Otherwise, the user may interpret the map in the wrong way without the correct base layer. Therefore, a digital map archive must be able to reproduce the original view of the dataset.

## 2.3 Evaluation of Geospatial Data Format for Archiving Purpose

This subchapter evaluates the most commonly used vector and raster formats by reviewing their ability to support long-term digital map archiving. In addition, an alternative format will be proposed and examined to determine whether it will have the potential ability to follow the principle and achieve the goals of digital maps preservation.

### 2.3.1 Vector Data Format – Shapefile

According to Pons & Masó-Pau (2016), 34 of 334 formats listed in the U.S. Library of Congress were categorized as geospatial-related formats. Among those data formats, shapefile is no doubt the most common vector data format in the GIS industry as ESRI has taken almost half of the world GIS market shares.

Released by ESRI in early 1990, the shapefile (SHP) format is a non-topographic geospatial vector data format, which can separately store the three basic vector feature types (points, polylines, and polygons) and can describe the feature information through an attribute table. In addition, the shapefile consists three mandatory binary files: Main file(.shp), Index file(.shx), and Dbase table(.dbf). A .shp file contains the geometric data of the shape feature; a .shx file contains provides an index of the feature geometry; a .dbf file stores the attribute information of the feature (ESRI, 1998). In addition to the three main files, it is common to see a .prj file in a shapefile which is used to store the projection information of the geospatial dataset.

The shapefile format has been widely used as the primary data format to archive and publish geospatial data in various archiving projects for different government agencies especially in the USA such as National Digital Information Infrastructure and Preservation Program (NDIPPP) by the Library of Congress. Under the NDIPPP, a Geospatial Multistate Archive and


Figure 2-6. GeoMAPP Partners (North Carolina Center for Geographic Information and Analysis & North Carolina Department of Cultural Resources, 2011)

Preservation Partnership (GeoMAPP) had been formed by a variety of universities and state agencies such as North Carolina Center for Geographic Information and Analysis.

Despite the advantages such as fast drawing capability, readily available supportability and wide uses (Theobald, 2001), the SHP does have many drawbacks that make it unsuitable for future digital map archiving and distribution. Cepicky et al.,(2017) listed several fatal disadvantages of the shapefile format such as no coordinate reference system definition by default, multi-file format , 4GB size limit, non-topological format, and poor support for attribute data types.

Since geospatial data must be able to be understood by users, metadata is equally important as geospatial data. Guptill (1999) explained the importance and standardization of metadata in his paper, "*standardized metadata elements provide a means to document datasets within an organization, to contribute to catalogues of data that helps individuals find and use existing data, and to help users understand the contents of datasets that they receive from others*", "*metadata provides descriptive information about the producer, content, quality, condition, and other characteristics of a given item.*" According to FGDC (2011), geospatial metadata may contain the following core components: metadata record information, identification information, constrain information, data quality information, maintenance information, spatial representation, reference system information, content information, symbology information, distribution information, metadata extension information, and application schema information.

Regarding of the support of metadata information, the SHP, by default, does not have the ability to store metadata. The metadata of the dataset is usually provided in a separate file such as XML or PDF. For example, Missouri Spatial Data Information Service (MSDIS), a partner in GeoMAPP, delivers the metadata information of a dataset in a XML link on the download page of the dataset (Red Rectangle Mark in Figure 2-7).

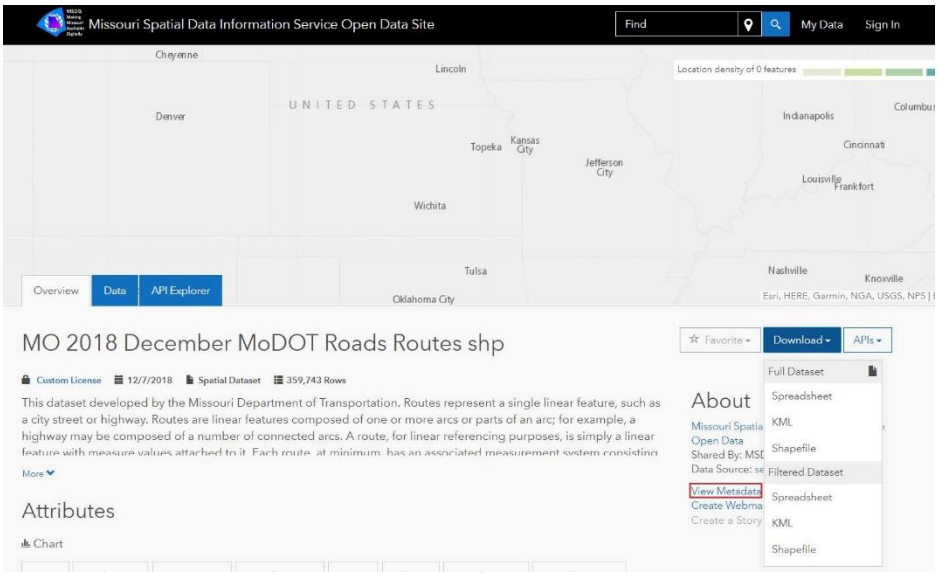



Figure 2-7. Support of Metadata on Geoportal of MSDIS website

In addition, the SHP does not support storing styling information such as symbology and feature labels within its file structure. For ArcGIS users, the style they create for a feature in ArcGIS software can be stored in a layer file(.lyr), which provides a link to the actual SHP file. Moreover, a .lyr file is a proprietary format that can only be read through ArcGIS software or service (ESRI, 2019). Since the layer file only store the link to the actual SHP, it usually causes a problem during data migration or sharing. Moreover, recreating map styling through original shapefile datasets is a consuming process for a GIS specialist.

From the view of data dissemination, due the multi-file characteristic of the SHP, the traditional approach is to include all relating data and information into one ZIP file and then share the ZIP file through file transfer protocol (FTP) service, which is still widely used by government agencies. Nowadays, the original data stored in shapefiles are common published or shared through Web Feature Service (WFS) hosted by government agencies.

### 2.3.2 Raster Data Format – GeoTIFF

As the inventor of GeoTIFF format, Ritter(1995) stated in his work, "*TIFF has emerged as one of the world's most popular raster file formats. But TIFF remains limited in cartographic applications, since no publicly available, stable structure for conveying geographic information presently exists in the public domain*". "*GeoTIFF format fully complies with the TIFF 6.0 specifications, and its extensions do not in any way go against the TIFF recommendation, nor do they limit the scope of taster data supported by TIFF*"(Mahammad & Ramakrishnan2003). In addition to the characters of TIFF format, GeoTIFF associates the cartographic information such as coordinate system and metadata to the original TIFF file in separated files.

From the view of archiving, GeoTIFF is similar to Shapefile in that they are both multi-file formats. Based on the principle of archiving digital maps, multi-file format has the potential to cause data loss during the data migration and dissemination process. In addition, since TIFF is a 32bit data format like SHP, the size limit of TIFF file is 4GB as well. However, as GeoTIFF is widely accepted by the GIS world, a majority of government agencies are still using it for the preservation of digital maps. For example, Swiss Federal Archives (SFA) and the Federal Office of Topography (swisstopo) are currently using the format to archive their images in project ellipse.

In terms of metadata, GeoTIFF has the ability to store the metadata in a separated XML file, which can be read by GIS application and users directly.

GeoTIFF, like SHP, is commonly disseminated in a compressed in ZIP file or shared through a web map service (WMS) such as Google map or Bing map.

### 2.3.3 Alternative Data Format for Future Digital Maps Archiving and Dissemination

As described in the latest GeoPackage Encoding Standard (OGC, 2018), "*GeoPackage is an open, standards-based, platform-independent, portable, self-describing, compact format for transferring geospatial information. It is a platform-independent SQLite database file that contain the GeoPackage and metadata."(See Figure 2-8)*

The core capabilities of GeoPackage allow it to store all three types of vector features, raster data at various scales, attributes, and extensions. The GeoPackage by default supports 12 different data types which are BOOLEAN, TINYINT, SMALLINT, MEDIUMINT, INTEGER, FLOAT, DOUBLE, TEXT, BOLB, <geometry_type_name>, DATE, and DATETIME. Additional data types can be added specifically through the GeoPackage Extension Mechanism.

*"A **GeoPackage** MAY be "empty" (contain user data table(s) for vector features, non-spatial attributes, and/or tile matrix pyramids with no row record content) or contain one or many vector feature type records and /or one or many tile matrix pyramid tile images. GeoPackage metadata CAN describe GeoPackage data contents and identify external data synchronization sources and targets. A GeoPackage MAY contain spatial indexes on feature geometries and SQL triggers to maintain indexes and enforce content constraints."(OGC, 2018)*

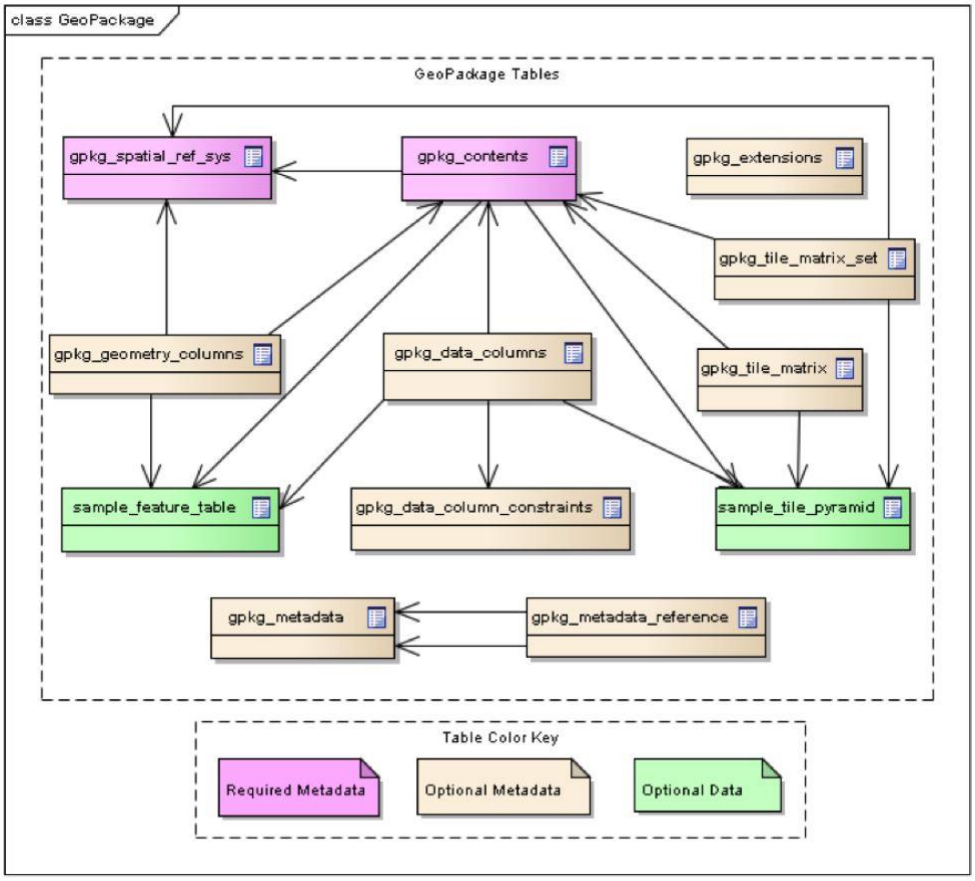



Figure 2-8. GeoPackage Tables Overview (OGC, 2018)

Figure 2-8 shows the default tables for a standard GeoPackage. Two general tables are gpkg_spaital_ref_sys and gpkg_content. The table, gpkg_spaital_ref_sys, can stores multiple coordinate systems that the vector and raster data uses. The gpkg_content table is used to list all geospatial contents with their data type, description, last change time, bounding box values, and spatial reference system ID. *(OGC, 2018)*

SHP and GeoTIFF were invented two decades ago, but GeoPackage (.gpkg) is a young geospatial data format which was released in 2014. The statement above has already shown some of the advantage of GeoPackage over the current most common vector and raster data formats.

Compared to SHP and GeoTIFF, which both are binary data formats, GeoPackage, in nature, is an ASCII SQLite based database container. Jobst and Gartner (2012) stated, "*While a binary format is a direct machine code, which cannot be humanly read, the ascii format can be directly read with any text editor. Thus the ascii format should be favored in terms of cartographic heritage.*" The nature of ASCII and SQLite format allows GeoPackage to be a self-contained geospatial data format, which can be accessed with minimal effort from a GIS application and operating system. In addition, it also makes GeoPackage a single file format that significantly reduces the workload of data packaging and preparation and the risk of data loss during the data migration and dissemination process.

Furthermore, both SHP and GeoTIFF were designed into the 32bit data format, which supports storing maximum 4GB data. With the increasing data size in modern environment, 4GB storage room seems insufficient. Meanwhile, GeoPackage, as a 64bit supported data format, allows maximum 150 TB storage size, which is adequate not only for today's usage but also for the future.

Since GeoPackage is new data format, there is not much research relating to using it for archiving purpose and dissemination. However, there is some research relating the use of GeoPackage on mobile applications. In a recent study by Rashinan et al.,(2016), they conducted an outdoor utility mapping project by using GeoPackage data format to store and distribute the data collected by a mobile GIS device. The results from their experiment revealed that the GeoPackage format had allowed them to seamlessly read and write feature data on a mobile device in an offline work environment. In terms of the raster data (MBTiles) they were using, GeoPackage also showed the advantage of supporting multiple tiles in one single file, while MBTiles can only have one tileset in a file within a specific projection. Another study by Bogossian et al.,(2014) presented "*a hybrid architecture for mobile geographic data acquisition and validation system that can operate online as well as offline."* It proved that GeoPackage, while supporting both vector data and tiled raster data, can be used as an "*interoperable file between SDI and the mobile systems."*

In term of metadata, the GeoPackage format, as a single file format and SQL container, has a natural advantage over SHP and GeoTIFF, in that it can integrate metadata as a table without saving the metadata as a separated XML file. Based on the OGC GeoPackage Encoding Standard (2018), the GeoPackage contains two tables, gpkg_metadata and gpkg_metadata_reference, that allows it to store metadata in MIME encoding. "*These tables are intended to provide the support necessary to implement the hierarchical metadata models as defined in ISO 19115"* (OGC, 2018). However, as the document stated, "*There is no GeoPackage requirement that such metadata be provided or that defined metadata be structured in a hierarchical fashion. This extension simply provides a mechanism for storing this information*." Based on the argument above, the first research question is raised, "Is it possible to embed relevant metadata for future usage in digital maps using GeoPackage? "

One important factor of archiving digital maps is the ability to preserve the map style information such as symbology and labeling within the data format. However, there is no information provided in OGC Encoding Standard regarding storing the map style.

In addition, a proper method of distributing digital maps should be considered as equally important as the archiving process. In Li et al.,'s research (2017), they illustrated the importance of displaying maps with the interactivity to access geographical features. Different than the traditional map service provided by Google Map or Bing Map, digital maps from mapping agencies such as USGS and BEV usually involves both vector data and raster with certain styling. Thus, the distribution of digital maps through a web service requires the functions of both WMS and WFS. As an emerging technology, "*vector tiles are packets of geographic data, packaged into tiles for transfer over the web. They can be used for delivering styled web maps with vector map data. Unlike raster tiled web maps, the server return vector map data, which has been clipped to the boundaries of each tile, instead of a pre-rendered map image*" Ordnance Survey (2018).

Thus, this brings the second research questions, "Is it possible to embed styling from GeoPackage when distributing digital maps that use vector-tile format?"

Python, as an open source and cross-platform programming language, has gradually became the most used programming language in the GIS community. Python is easy to learn and to deploy on multiple operating systems without installing additional framework or extensions. Moreover, Python has been widely supported by the majority of GIS applications on the market such as ArcGIS, QGIS, and GRASS. Also, Python, as a glue language, has a strong ability to integrate different programming languages and to access various API and libraries. (Altaweel, 2017 & ESRI, 2018) Considering that archiving digital maps is a time-consuming process and Python 3 has a module in its Standard library to manage SQLite database, the third research question is being brought up, "Is it possible to create Python scripts that partially automate the procedures of embedding relevant information into GeoPackage?"

# 3. Methodology and Case Study

## 3.1 Case Study Background and Scenario

The Austrian National Mapping Agency (BEV) shares its geodata mainly in shapefile format and GeoTIFF format and hosts the web service of the Austrian Cartographic Model with national map styling. However, apart from viewing the map in different scales and with a general measurement tool, the BEV web map service does not allow too much user interaction such as viewing feature attribute. Also, the spatial data provided by BEV does include the metadata as PDF in the folder but lacks style information and certain layers needed to reproduce a similar Austrian map with national map styling.

The current situation of the Austrian Cartographic Model with national map styling from BEV brings up several problems concerning the preservation and dissemination of digital maps. In one example, the scenario is that a GIS specialist intends to archive the national maps but he/she can only obtain the shapefile and raster data from BEV. Should the GIS specialist create the map from scratch? A second possible scenario may happen to a regular Austrian citizen who wants to have a digital collections of the Austrian Cartographic Model with national map styling. Does the citizen have to purchase the digital maps from BEV? A third possible scenario can happens when there is an emergency event such as the recent Etna Volcano eruption in Sicily, Italy. Is it possible to make and share the maps in a short time period?

This thesis presents a method to solve the potential problems in these scenarios by archiving the 1:1 Million Scale Maps from BEV with GeoPackage and distributing the digital maps using GeoServer.

## 3.2 Research Design

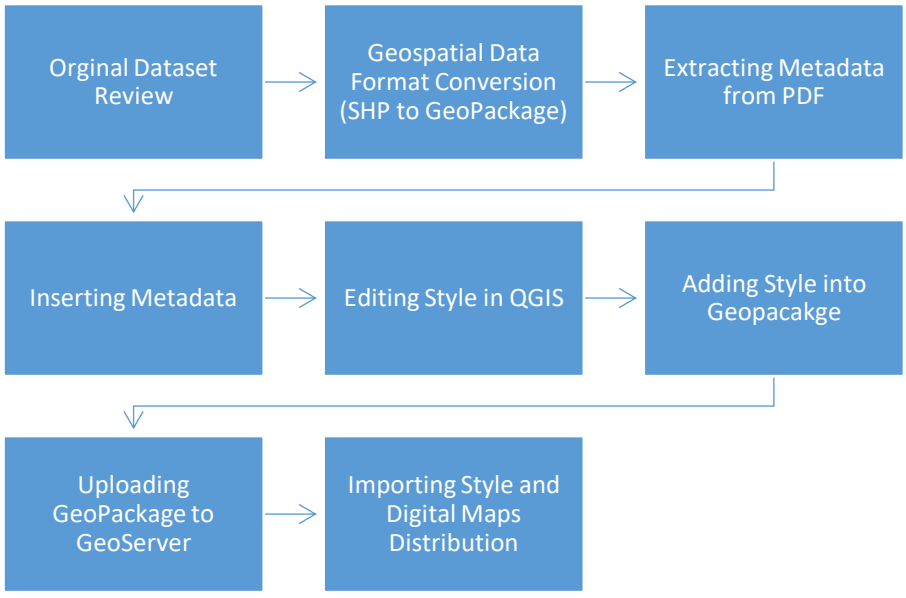



Figure 3-1. Workflow Overview

### 3.2.1 Overview of Approach

In order to solve the problems in the above scenarios and to answer the research questions, the workflow (see Figure 3-1) is designed that follows the principles of the preservation and dissemination of digital maps presented in the Chapter Two. The workflow includes eight stages and each of them is explained in detail in the following subchapters.

### 3.2.2 Operating System and Software Environment

Considering the principle of archiving digital maps and the challenges, this thesis aims to provide a simple and cross-platform solution. All the software and programming language involved in this approach are listed in Table 1, which are supported in both main PC operating systems, Windows and MacOS. In addition, all the software and programming language are free, open-source and platform-independent.

| WORKFLOW | SOFTWARE/PROGRAMMING LANGUAGE |
|---|---|
| Original Dataset Review | PDF Reader |
| Geospatial Data Format Conversion | QGIS 3.4.1 |
| Extracting Metadata From PDF | PDF Reader/Python 3 IDLE/SQLite |
| Inserting Metadata | Python 3 IDLE/SQLite |
| Editing Style in QGIS | QGIS 3.4.1/Color Picker |
| Adding Style into GeoPackage | QGIS 3.4.1 |
| Uploading GeoPackage | GeoServer 2.15 |
| Importing Style and Digital Maps Distribution | GeoServer 2.15 |

Table 1. Software and Programming Language Requirement

#### 3.2.2.1 Python Environment

In order to run the Python script created for automating part of these workflow procedures, it is required to install specific modules and dependencies which are free to download and that work in both Windows and MacOS operating system. The Table 2 displays all built-in libraries and modules required to run the Python script while Table 3 shows all external modules and dependencies required.

| Built-in Libraries and Modules | Functionality |
|---|---|
| sys, os, io | Access operating system functionality |
| shutil | Support file copying and deleting in OS |
| sqlite3 | Access the database using SQL language |
| re | Provide expression matching operations |
| csv | Import and export CSV file |
| xml | Process XML file |

Table 2. Built-in Modules used in Python script

| External Modules and Dependencies | Functionality |
|---|---|
| PySimpleGUI | Create custom GUI Interface |
| DBManager | Support connections to GeoPackage |
| Pandas | Access and manage structured and time series data. |
| PDFMiner | Extract text from PDF document |
| Tabula-py | Access tables in PDF document and convert into DataFrame supported by Pandas |
| datefinder | Extract dates from text |
| Datatime | Manipulate time data format |
| urllib3 | Dependency of Tabula-py |
| distro | Dependency of Tabula-py |
| Java SE Development Kit(Java JDK) | Support functions in Tabula-py module |

Table 3. External Modules and Dependencies required by Python script

To make the approach more user-friendly and avoid the hassle of installing the external modules for users, two separate command line scripts are created for: Windows system (.bat) and MacOS system (.command), to automate installation of the external modules and dependencies for users. Both scripts have been tested in each operating system and work properly. The code of each command line scripts are provided in the appendix (see Annex 1 & Annex 2).

### 3.2.3 Original Dataset Review

As the first stage in the workflow, this stage aims to explore the structure of the original dataset and identify the content that needs to be converted into GeoPackage format for archiving purpose. The geospatial dataset (KM1000-V) used for the 1:1 Million Austrian Cartographic Model is obtained from the BEV official website. However, this dataset contains only the vector


Figures 3-2. Original Dataset Overview

data for the national map. The unzipped dataset folder includes 18 SHPs, 2 PDF files, 3 DBF files, 1 TXT file, 78 files in total (see Figure 3-2). The size of the unzipped folder is 6.51 MB.

Except for the SHP files, the majority of Metadata relating to the geospatial dataset is stored between pages 22 to 33 of the PDF named BEV_S_KM50_KM250_KM500_KM1000_V_V1.5 (see example in Figure 3-3). The TXT file contains the general information in the dataset and the creation date of the datasets. All the metadata stated above is subjected to be extracted and transferred into a GeoPackage later.



## 2.4 Kartographisches Modell 1:1 Million – Vektor (KM1000-V)

### 2.4.1 General structure

#### 2.4.1.1 About KM1000-V

Data of KM1000-V is the Austrian part of EuroGlobalMap (EGM) the pan-European vector dataset at small scale. EGM Database is intended to be used in map scale 1:1 000 000.
Detailed specifications are described in ***EGMspec3-0se.pdf***. This document is a summary of the most relevant specifications.

#### 2.4.1.2 Data format and file table

Data of KM1000-V is stored in these files:

| Shape Files | Description | Type |
|---|---|---|
| KM1000_AIRFLD_POINT.shp | Airport / Airfield | Point |
| KM1000_DAM_LINE.shp | Dam/ Weir | Line |
| KM1000_ELEV_POINT.shp | Height point | Point |
| KM1000_GLACIER_AREA.shp | Glacier | Area |
| KM1000_ISLAND_AREA.shp | Island | Area |
| KM1000_LAKE_AREA.shp | Lake | Area |
| KM1000_NAME_POINT.shp | Named location | Point |
| KM1000_POLBND_AREA.shp | Administrative area | Area |
| KM1000_POLBND_LINE.shp | Administrative boundary | Line |
| KM1000_RAILRD_NODE.shp | Railway station | Point (Node) |
| KM1000_RAILRD_LINE.shp | Railway | Line |
| KM1000_RESERVOIR_AREA.shp | Reservoir | Area |
| KM1000_ROAD_LINE.shp | Road | Line |
| KM1000_BUILTUP_AREA.shp | Built-up area | Area |
| KM1000_BUILTUP_POINT.shp | Built-up point | Point |
| KM1000_SPRING_NODE.shp | Spring/ Water hole (connected) | Point (Node) |
| KM1000_WATRCRS_AREA.shp | Watercourse | Area |
| KM1000_WATRCRS_LINE.shp | Watercourse | Line |

| Info Tables | Description |
|---|---|
| ADMIN_ISN.dbf | This table includes the names of the administrative hierarchy levels |
| EGM_CHR.dbf | This table describes the national character sets used for each language. |
| SHN_NAM.dbf | The table includes the names of the units of all administrative levels. |

Figure 3-3. Example of Metadata in BEV PDF Page 22

### 3.2.4 Geospatial Data Format Conversion

After reviewing the original dataset, the second stage is to pack all the SHP files into an empty GeoPackage. This stage involves the use of one QGIS toolbox, Package Layer (see Figure 3-4). In this stage, all the SHP files were first imported into QGIS as feature layers, then the toolbox was used to pack feature layers into one GeoPackage. When using the toolbox, it is important to change the encoding method from ISO 8859-1 (Latin-1) to ISO 10646(UTF-8) to avoid the issue of displaying German umlauts characters. This issue is explained later in detail in Chapter 5.

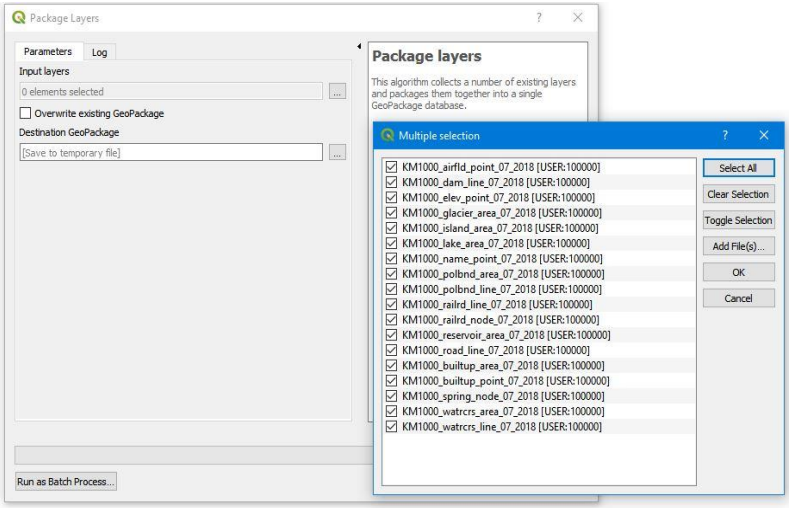



Figure 3-4. Package Layer Tool Box in QIS

The output of this stage is a GeoPackage with all geospatial information from the original datasets. Each layer is saved as four spatial index tables and one attribute table. The related spatial information from each layer is added to gpkg_contents, gpkg_extensions, gpkg_geometry_columns, and gpkg_spatial_ref_sys table. The output result can be checked in either QGIS or any open-source SQL database application such as DB Brower for SQLite.

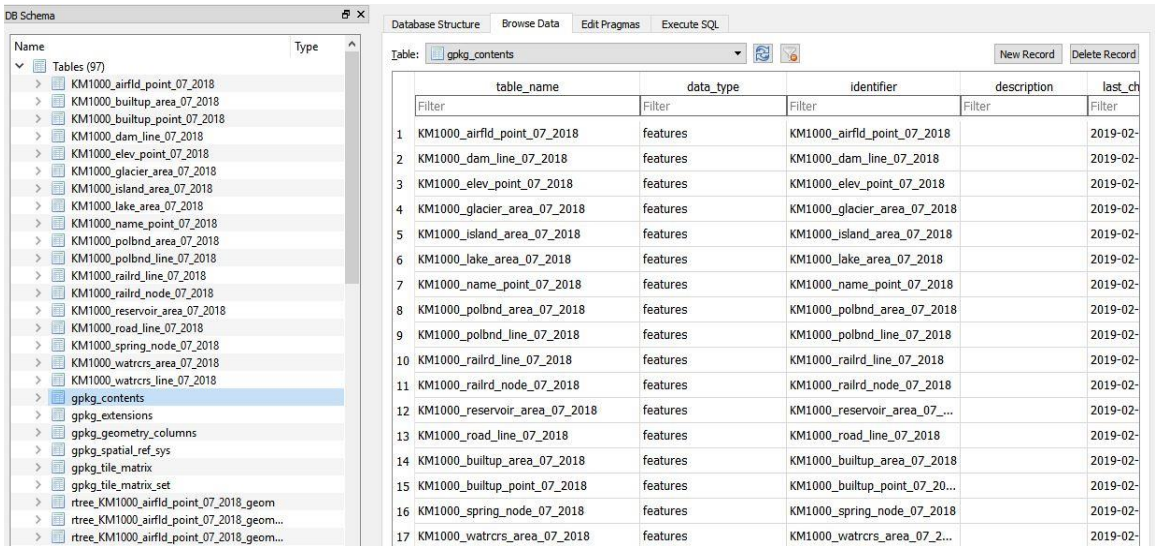



Figure 3-5. gpkg_contents table in output GeoPackage using DB Brower for SQLite

### 3.2.5 Extracting Metadata from PDF

This stage involves creating Python script to extract text and tables from the PDF file mentioned in the first stage.

The first part of the source code below is used to extract the text information from PDF and export into a temporary TXT file. The second part of the source code extracts tables from each page in the PDF file, and exports to temporary CSV files. The reason why the temporary output format is in CSV format is to keep the original table format so it can be inserted into SQL schema later in the fourth stage.

```
##Convert PDF to Text
##Source Code: http://stanford.edu/~mgorkove/cgi-bin/rpython_tutorials/Using%20Python%20to%20Convert%20PDFs%20to%20Text%20Files.php
def convert(fname, pages=None):
    if not pages:
        pagenums = set()
    else:
        pagenums = set(pages)

    codec = 'utf-8'
    output = StringIO()
    manager = PDFResourceManager()
    converter = TextConverter(manager, output, laparams=LAParams())
    interpreter = PDFPageInterpreter(manager, converter)

    infile = open(fname, 'rb')
    for page in PDFPage.get_pages(infile, pagenums):
        interpreter.process_page(page)
    infile.close()
    converter.close()
    text = output.getvalue()
    output.close
    return text

output = convert(BEVPDF, pages=[21])
print (output)

file = open("PDFPage22.txt", "w", encoding='utf-8')
file.write(output)
file.close()
print ("FINISHED Converting pdf to txt")
```

```
##Convert tables in PDF to CSV
tabula.convert_into(BEVPDF,"DataFormatFileTable.csv",encoding='utf-8',multiple_tables= True,output_format='data_format',pages="22")
tabula.convert_into(BEVPDF,"tablepage23.csv",encoding='utf-8',guess = False, multiple_tables= True,output_format='data_format',pages="23")
tabula.convert_into(BEVPDF,"tablepage24.csv",encoding='utf-8',guess = True, lattice = True, multiple_tables= True,output_format='data_format',pages="24")
tabula.convert_into(BEVPDF,"tablepage25.csv",encoding='utf-8',guess = True, lattice = True, multiple_tables= True,output_format='data_format',pages="25")
```

```
tabula.convert_into(BEVPDF,"tablepage2627.csv",encoding='utf-8',guess = True, lattice = True, multiple_tables= True,output_format='data_format',pages="26,27")
tabula.convert_into(BEVPDF,"tablepage28.csv",encoding='utf-8',guess = True, lattice = True, multiple_tables= True,output_format='data_format',pages="28")
tabula.convert_into(BEVPDF,"tablepage2930.csv",encoding='utf-8',guess = True, lattice = True, multiple_tables= True,output_format='data_format',pages="29,30")
tabula.convert_into(BEVPDF,"tablepage31.csv",encoding='utf-8',guess = True, lattice = True, multiple_tables= True,output_format='data_format',pages="31")
tabula.convert_into(BEVPDF,"tablepage32.csv",encoding='utf-8',guess = True, lattice = True, multiple_tables= True,output_format='data_format',pages="32")
tabula.convert_into(BEVPDF,"tablepage33.csv",encoding='utf-8',guess = True, lattice = True, multiple_tables= True,output_format='data_format',pages="33")
print ("FINISHED EXTRACT Table from PDF")
```

### 3.2.6 Inserting Metadata

With the related metadata extracted in the third stage, this stage involves inserting metadata in text and table into the GeoPackage using Python and SQL code. In addition, this stage contains multiple steps to insert different parts of the metadata into the GeoPackage. The temporary TXT and CSV files created in the third stage will be deleted once the insertion process is complete.

First, a connection to the GeoPackage needs to be established in order to insert the metadata later. The code below creates an interface which allows users to choose the GeoPackage they want to connect and sets up a connection.

```
GeoPackage = sg.PopupGetFile('Please Select the GeoPackage:')
if GeoPackage != "":
    conn = sqlite3.connect(GeoPackage)
    sg.Popup(GeoPackage, 'Database Connection Successful')
else:
    sg.Popup(GeoPackage, 'Database Connection Failed','Program Stopped')
    sys.exit("Error")

print ("Database Connection Successful")

cursor = conn.cursor()
```

Second, to insert the general information from the whole KM1000-V dataset, two standard metadata tables in MIME encoding structure, gpkg_metadata and gpkg_metadata_reference, are created based on the OGC GeoPackage Encoding Standard (OGC, 2018).

```
##Create gpkg_metadata table
cursor.execute('DROP TABLE IF EXISTS gpkg_metadata')
cursor.execute('''''CREATE TABLE gpkg_metadata (
  id INTEGER CONSTRAINT m_pk PRIMARY KEY ASC NOT NULL,
  md_scope TEXT NOT NULL DEFAULT 'dataset',
  md_standard_uri TEXT NOT NULL,
  mime_type TEXT NOT NULL DEFAULT 'text/xml',
  metadata TEXT NOT NULL DEFAULT ''
```

```
);''')
conn.commit()
##Create gpkg_metadata_reference table
cursor.execute('DROP TABLE IF EXISTS gpkg_metadata_reference')
cursor.execute('''''CREATE TABLE gpkg_metadata_reference (
  reference_scope TEXT NOT NULL,
  table_name TEXT,
  column_name TEXT,
  row_id_value INTEGER,
  timestamp DATETIME NOT NULL DEFAULT (strftime('%Y-%m-%dT%H:%M:%fZ','now')),
  md_file_id INTEGER NOT NULL,
  md_parent_id INTEGER,
  CONSTRAINT crmr_mfi_fk FOREIGN KEY (md_file_id) REFERENCES gpkg_metadata(id),
  CONSTRAINT crmr_mpi_fk FOREIGN KEY (md_parent_id) REFERENCES gpkg_metadata(id)
);''')
conn.commit()
```

The following third step is to insert the general information and structure metadata from the KM1000-V dataset. The general structure from page 22 of the PDF file and the general information in the TXT file named Aktualitaetsstand are extracted from the text file converted earlier in the third stage, and then inserted into the gpkg_metadata table using the Python and SQL code below.

```
##Path Location of Aktualitaetsstand
LatestInfo = sg.PopupGetFile('Please Select Aktualitaetsstand.txt file in Unzipped KM-1000V Folder:')
if LatestInfo !="":
    sg.Popup(LatestInfo, 'Import Successful')
else:
    sg.Popup(LatestInfo, 'Import Failed','Program Stopped')
    sys.exit("Error")

lines = []

file = open(LatestInfo, 'rt',encoding="latin-1")

for line in file:
    lines.append(line)

ReleaseDate = lines[3] + lines[5]
print (ReleaseDate)


##Extract General Structure Information of KM-1000
ExtractGeneralStructure = "PDFPage22.txt"
file = open(ExtractGeneralStructure, 'r',encoding='utf-8')
content = file.read()
GeneralStructurePattern = r'2.4.1.1  About KM1000-V (.*\n.*\n.*\n.*\n.*) .*'
test = re.search(GeneralStructurePattern, content, re.MULTILINE)
GeneralStructure = str(test.group())
GeneralInformation = ReleaseDate + GeneralStructure
print (GeneralInformation)

##Append general metadata infotmation to gpkg
metadata = [1,'dataset','NA', 'text', GeneralInformation]
print (metadata)
cursor.execute('insert into gpkg_metadata values (?,?,?,?,?)', metadata)
conn.commit()
print ('''''FINISHED Extracting Text from Aktualiaetsstand and PDF Page 22
```

```
FINISHED Adding general metadata infotmation to gpkg''')

file.close()
if os.path.exists("PDFPage22.txt"):
  os.remove("PDFPage22.txt")
else:
  print("The file does not exist")
```

Next, the fourth step is to insert the layer description metadata and the description of layer attribute metadata into the GeoPackage. While the standard metadata table provided by OCG standard encoding is not sufficient to cover the layer description metadata and the description of layer attribute metadata (see example in Figure 3-6), two metadata tables are created for each vector layer in the GeoPackage. KM1000_polbnd_area layer is used as an example to explain the process in the fourth step.

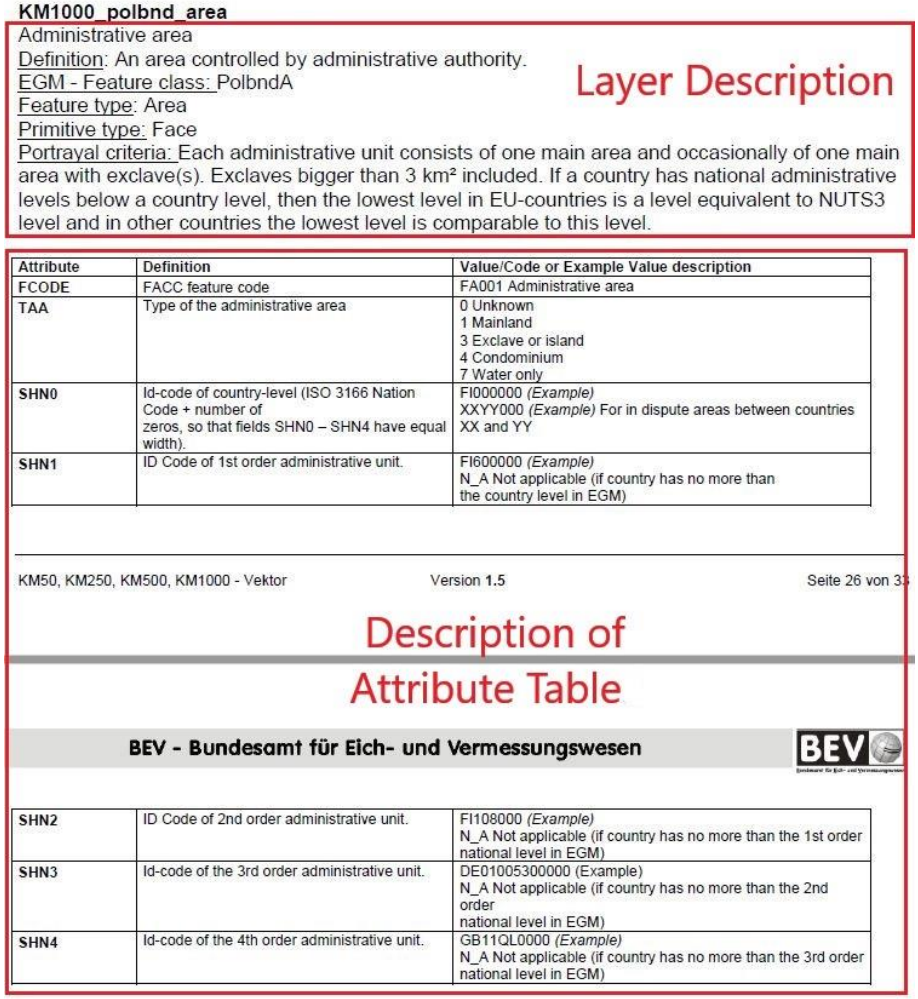



Figure 3-6. Metadata Overview of KM1000_polbnd_area Layer in BEV PDF

The python code below shows how to insert the layer description and the description of attribute table of KM1000_polbnd_area layer. The layer description is stored as a table named KM1000_polbnd_area_07_2018_description_of_attributes in the GeoPackage, while the description of attribute table is stored in KM1000_polbnd_area_07_2018_description. The

information was extracted from the temporary TEXT file and CSV file created in the third stage. All the temporary CSV and TXT files are removed when the insertion process is finished.

```
##add description of layer to gpkg for KM1000_polbnd_area
FeatureClassName = "KM1000_polbnd_area"
Definition = "An area controlled by administrative authority."
EGM_Feature_Class = "PolbndA"
FeatureType = "Area"
PrimitiveType = "Face"
PortrayalCriteria = "Each administrative unit consists of one main area and occasionall
y of one main area with exclave(s). Exclaves bigger than 3 km² included. If a country h
as national administrative levels below a country level, then the lowest level in EU-
countries is a level equivalent to NUTS3 level and in other countries the lowest level
is comparable to this level."
attributeTable = "KM1000_polbnd_area_07_2018_description_of_attributes"
table = [1,FeatureClassName, Definition, EGM_Feature_Class, FeatureType, PrimitiveType,
 PortrayalCriteria]
cursor.execute('DROP TABLE IF EXISTS KM1000_polbnd_area_07_2018_description')
cursor.execute('''''CREATE TABLE KM1000_polbnd_area_07_2018_description (
        id INTEGER CONSTRAINT m_pk PRIMARY KEY ASC NOT NULL,
        featureClassName TEXT NOT NULL,
        definition TEXT NOT NULL,
        EGMFeatureClass TEXT NOT NULL,
        featureType TEXT NOT NULL,
        primitiveType TEXT NOT NULL,
        portrayalCriteria TEXT NOT NULL
        );''')
cursor.execute('INSERT INTO KM1000_polbnd_area_07_2018_description VALUES (?,?,?,?,?,?,
?)', table)
conn.commit()
print ("FINISHED Inserting description for KM1000_polbnd_area")

##add description of attribute table to gpkg for KM1000_polbnd_area
skiprows = list(range(9)) + list(range(17,26))
DescriptionOfAttributes = pd.read_csv("tablepage2627.csv",skiprows = skiprows, usecols
= [0,2,3],encoding='latin1')
cursor.execute('DROP TABLE IF EXISTS KM1000_polbnd_area_07_2018_description_of_attribut
es')
DescriptionOfAttributes.to_sql("KM1000_polbnd_area_07_2018_description_of_attributes",c
onn, if_exists='append',index=False)
conn.commit()
print ("FINISHED Inserting description of attributes for KM1000_polbnd_area")
```

In addition to the metadata provided in the PDF file and TXT file by BEV, it is necessary to consider the compatibility of GeoPackage with INSPIRE metadata. According to INSPIRE policy (2007), Austria, as a member state in European Commission, is obligated to guarantee the creation and updating of metadata for the geospatial dataset and service. In the latest INSPIRE technical guidelines (2017), the discoverability of datasets and the Spatial Data Services is based on "*the data and service providers describing their resources using the metadata elements according to rules mandated by the INSPIRE regulations, and on the other hand, the Discovery Services providing online access to query the provided metadata.*" INSPIRE metadata is encoded in XML schema following ISO standards 19115/19119/19139. *"The abstract standards [ISO 19115] and [ISO 19119] provide a structural model and specify the content of the set of metadata elements used in this specification, but they do not specify the encodings of those elements. The [ISO 19139] specifies an XML encoding of [ISO 19115] elements, but not for the service-specific metadata elements contained in [ISO 19119]"* (INSPIRE MIG, 2017). Normally, an INSPIRE

metadata XML file includes three sections: general requirements (file identifier, metadata language, metadata point of contact, and metadata date), identification info section (resource title, resource abstract, the responsible organization and point of contact for the described resource, temporal reference, using keywords, limitations on public access, conditions applying to the access and use, and geographic bounding box), and data quality info section (conformity). The EU member states are obligated to adapt INSPIRE metadata standard to ensure compatibility and usability of their SDI.

Thus, this addition step aims to insert INSPIRE XML metadata to the GeoPackage. The INSPIRE metadata of KM1000_polbnd_line layer in XML format was provided by Dr. Markus Jobst at BEV. The following Python code was created to insert the metadata in XML format to the gpkg_metadata and gpkg_metadata_reference tables in GeoPackage. Two XML records are made by the Python code. One of them keeps the original format of the XML file (see Figure 3-7). The other one removes the elements in XML and keeps only the child tag and element so that the XML file is more readable to non-technical personnel.

```
1.  ##insert KM1000_polbnd_line XML to gpkg
2.  ##polbnd_line_xml = "KM1000AU__d2b8d67f-737c-4d49-b220-ca0ef422197d.xml"
3.  polbnd_line_xml = sg.PopupGetFile('Please Select Downloaded XML File:')
4.  if LatestInfo !="":
5.      sg.Popup(polbnd_line_xml, 'Import Successful')
6.  else:
7.      sg.Popup(polbnd_line_xml, 'Import Failed','Program Stopped')
8.      sys.exit("Error")
9.
10. polbnd_line_xml_copy = "polbnd_line_xml_copy.xml"
11. copyfile(polbnd_line_xml, polbnd_line_xml_copy)
12.
13. xmlformatpreserved = xml.dom.minidom.parse(polbnd_line_xml_copy)
14. xmlformatpreserved = xmlformatpreserved.toprettyxml()
15. xmlformatpreserved = str(xmlformatpreserved)
16.
17.
18. metadata = [2,'feature','http://www.isotc211.org/2005/gmd', 'xml', xmlformatpreserved]

19. cursor.execute('insert into gpkg_metadata values (?,?,?,?,?)', metadata)
20. conn.commit()
21. print ("Finishing inserting xml to GeoPackage")
22.
23. timestamp = datetime.datetime.now()
24. metadata_reference = ['table','KM1000_POLBND_LINE_07_2018', '', '',
25.                       timestamp, 2, 1]
26. cursor.execute('insert into gpkg_metadata_reference values (?,?,?,?,?,?,?)', metadata_r
    eference)
27. conn.commit()
28.
29. def replace_in_config(old, new):
30.     with open(polbnd_line_xml_copy, 'r') as f:
31.         text = f.read()
32.
33.     with open(polbnd_line_xml_copy, 'w') as f:
34.         f.write(text.replace(old, new))
35.
36. replace_in_config('gco:', '')
37. replace_in_config('gmd:', '')
```

```
38.
39.
40. tree = ElementTree.parse(polbnd_line_xml_copy)
41. root = tree.getroot()
42.
43. extractxml = ""
44.
45. for child in root.iter():
46.     tagstring = str(child.tag)
47.     textstring = str(child.text)
48.     extractxml += tagstring + textstring
49.
50. extractxml = str(extractxml)
51.
52. metadata = [3,'feature','http://www.isotc211.org/2005/gmd', 'text', extractxml]
53. cursor.execute('insert into gpkg_metadata values (?,?,?,?,?)', metadata)
54. conn.commit()
55.
56. timestamp = datetime.datetime.now()
57. metadata_reference = ['table','KM1000_POLBND_LINE_07_2018', '', '',
58.                       timestamp, 3, 1]
59. cursor.execute('insert into gpkg_metadata_reference values (?,?,?,?,?,?,?)', metadata_r
    eference)
60. conn.commit()
61.
62. print ("Finishing inserting xml to GeoPackage")
63. if os.path.exists("polbnd_line_xml_copy.xml"):
64.     os.remove("polbnd_line_xml_copy.xml")
65. else:
66.     print("The file does not exist")
```

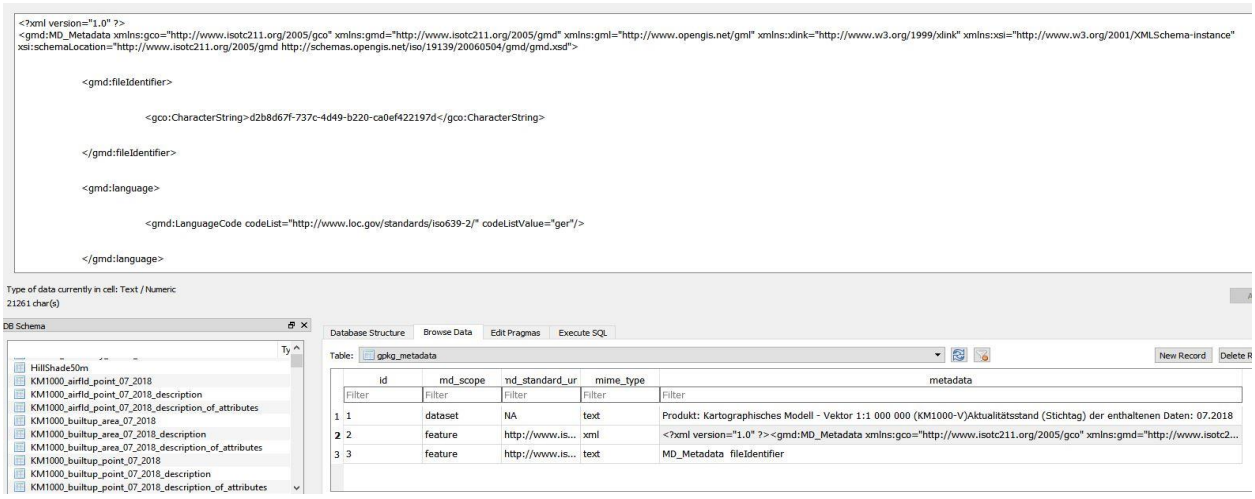



Figure 3-7. INSPIRE XML with Original Format in GeoPackage

### 3.2.7 Editing Style in QGIS

This stage aims to use QGIS software to create the style for the GeoPackage to the match the national map style of the Austria Cartographic Model at 1:1 million scale (see Figure 1-1).

The first step is to edit the symbology of vector layers displayed in the Austria Cartographic Model. A free color-picker tool is used to detect the color code of the vector feature in the national map. The symbology of each feature layer in the GeoPackage is edited or created to match the shape and color of features in the national map (see Figure 3-8).

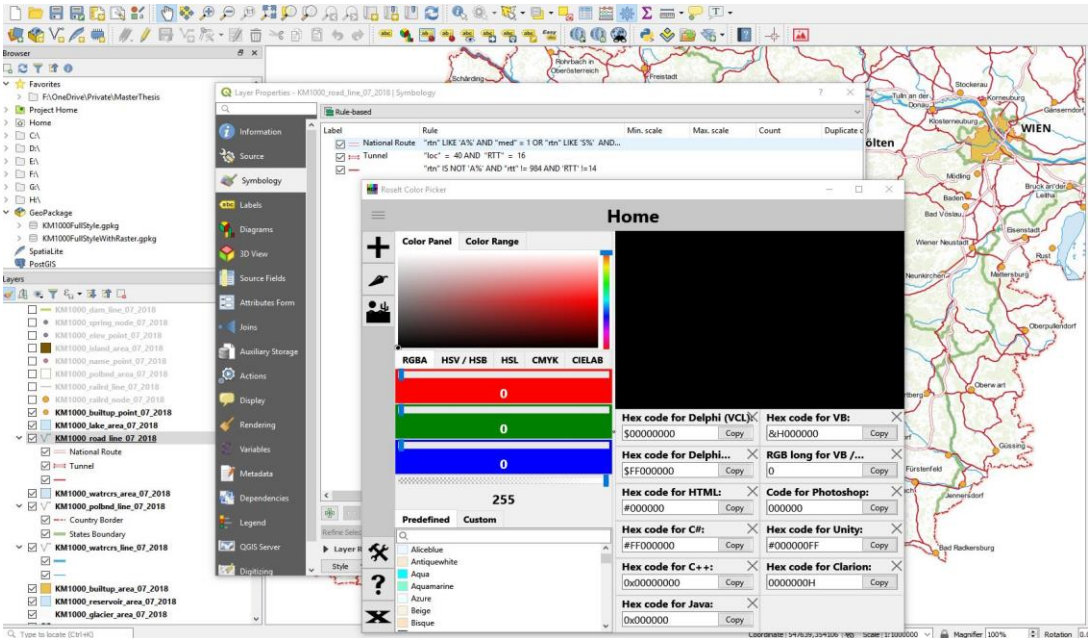



Figure 3-8. Style Editing Process in QGIS

Since the KM1000 folder does not provide data to make certain cartographic representations and the base map, additional data and processes are required to create the base map and a new layer is generated to display as cartographic representation (see Figure 3-9). To be more specific, the buffer layer, Austria_Boundary_Buffer_Effect, was created using the political



Figure 3-9. Additional Cartographic Representations created to match the National Map Style

boundary layer stored in GeoPackage and the symbology of the buffer layer was changed to the national map style. Moreover, the Austrian DEM dataset from BEV was used to generate the hillshade layer. The forest coverage layer from BEV was clipped by the buffer layer and its symbology has been changed to match the national map style. Furthermore, some features that were not displayed in the 1:1 million national map were removed from the attribute table. Finally, the features were labeled in accordance with the national map.

### 3.2.8 Adding Style into GeoPackage

The sixth stage is to insert the styles created in the fifth stage to the GeoPackage through QGIS. QGIS provides the function in layer property to save the vector layer style in GeoPackage (see Figure 3-10). This function generates a new table named layer_styles in QGIS and add the style of each vector layer in QML and SLD format to the layer_styles table.

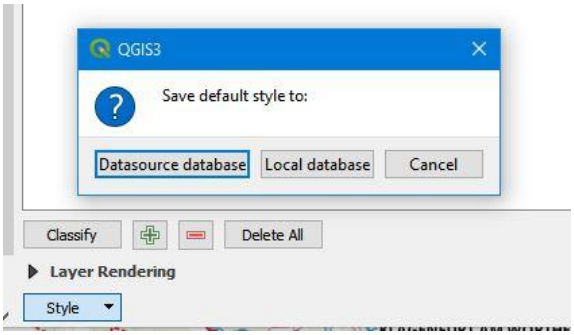



Figure 3-10. Add Layer Style to GeoPackage in QGIS

As described in Styled Layer Descriptor profile of the Web Map Service Implementation Specification (OGC, 2007), the SLD format is an non-proprietary XML schema for storing the style information of either vector or raster map layers and is widely used by WMS to style specific layers. Nonetheless, QGIS does not have a built-in function to either insert the SLD file of raster layer to GeoPackage or to export the SLD file separately. To solve this issue, the SLD4raster is needed from QGIS Python plugins repository and the SLD file of raster is manually inserted into the layer_styles table in GeoPackage.

This stage also marks the end of archiving process with GeoPackage.

### 3.2.9 Uploading GeoPackage to Geoserver

In this stage, all the map layers in GeoPackage are uploaded to GeoServer. The uploading process followed the standard operating procedure in GeoServer. First, a workspace was created and WMTS, WFS, WMS services were enabled in the settings. Next, the completed GeoPackage was added in Stores and each layer in Stores was published individually. During the publishing procedure, the Coordinate Reference System of each layer was changed from GeoServer default setting to EPSG: 3416, which is the Austria Lambert Project and European Terrestrial Reference System 1989, used by the original BEV dataset. Then, the parameter of Bounding Boxes of each layer was calculated from the Coordinate Reference Systems. Once all the layers in Stores were published, a Layer Groups was created to include all 12 layers used to display in the 1:1 million Austrian National Map. The 12 layers should follow the drawing order below so the final maps

can match the national map. When overlaying the layers in QGIS to match the national map, the order below needs to be reversed.

1. Austria_Boundary_Buffer_Effect
2. HillShade50m
3. KM500_R_WALD_07_2018
4. KM1000_glacier_area_07_2018
5. KM1000_reservoir_area_07_2018
6. KM1000_builtup_area_07_2018
7. KM1000_watrcrs_line_07_2018
8. KM1000_polbnd_line_07_2018
9. KM1000_watrcrs_area_07_2018
10. KM1000_road_line_07_2018
11. KM1000_lake_area_07_2018
12. KM1000_builtup_point_07_2018

### 3.2.10 Importing Style and Digital Maps Distribution

The final stage is about the dissemination of the digital maps. Instead of simply sharing the original geospatial data like BEV does, it is important to embed the style with the datasets so the digital map can be distributed properly. In this stage, the symbology of each layer stored in GeoPackage is uploaded to the Geoserver in SLD format and set as the default style for the corresponding layer (see Figure 3-11).

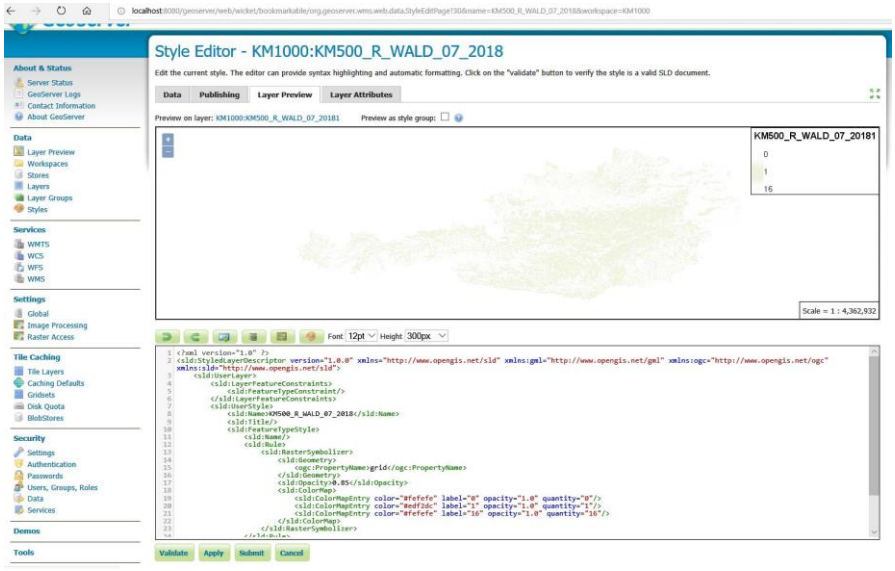



Figure 3-11. Uploaded style in SLD format in Geoserver Style Editor

Finally, in order to share the digital maps as vector tiles, the Vector Tiles Extension must be added to the GerServer directory. For now, the GeoServer supports sharing the digital maps in three vector tiles formats, which are GeoJSON vector tiles, MapBox vector tiles, and TopoJson Vector Tiles (See red marks in Figure 3-12). Sharing vector tiles is a new function supported only

by the latest version of GeoServer, released on February 18, 2019. Therefore, it has some bugs when using the style editor. The issue is discussed later in Chapter 5.

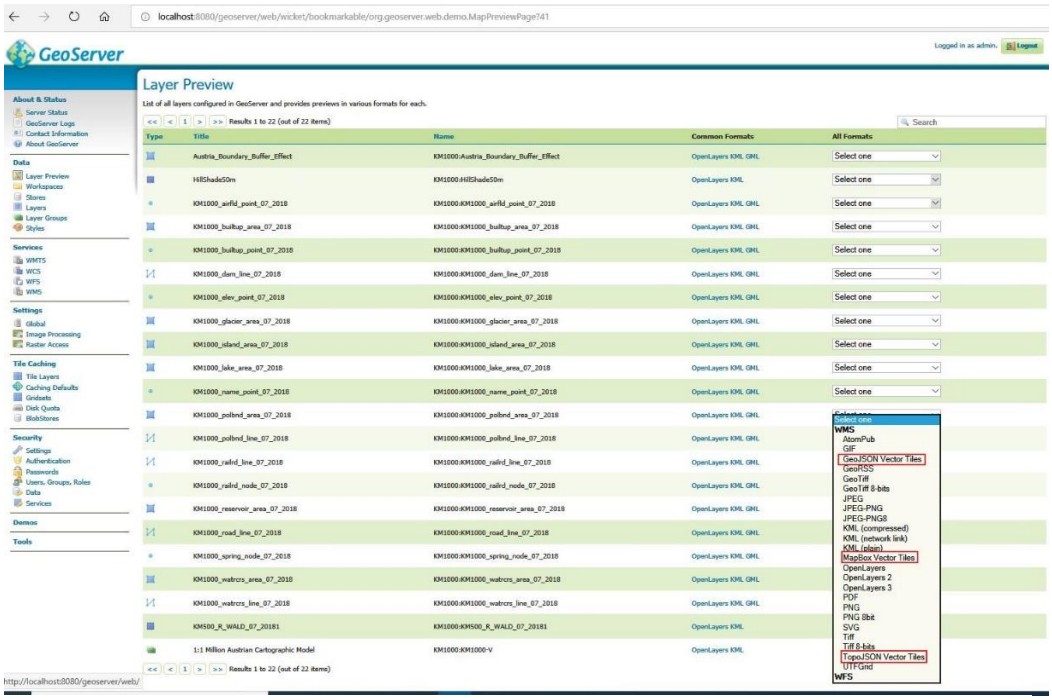


Figure 3-12. Sharing Digital Maps in Vector Tiles Format in GeoServer 2.15

# 4. Results

In general, the results obtained through this research method were successful and all three questions have been answered. This chapter presents the final results of the research in details. The issues occurred in the workflow are discussed in later chapter.

## 4.1 Archiving Digital Maps with GeoPackage

One of the hidden requirement of archiving digital maps is the size of final dataset. Thus, two GeoPackages, KM1000FullStyle.gpkg and KM1000FullStyleWithRaster.gpkg are generated to fulfill the different needs.

The KM1000FullStyle GeoPackage contains all 18 feature layers from the original dataset, all relating metadata information from the PDF and TXT file, INSPIRE XML metadata for the KM1000_polbnd_line_07_2018 layer, and style information in QML and SLD format. No feature has been removed from the original layer.

In addition to the content stored in KM1000FullStyle GeoPackage, the KM1000FullStyleWithRaster GeoPackage includes two raster layers (Forest Coverage and Hillshade) and one vector layer (Austria_Boundary_Buffer_Effect). In order to match the national map content, some features have been removed from the original layer.

Figure 4-1 indicates the size difference between of the datasets. While the original BEV dataset in the ZIP file is about 4MB, the KM1000Fullstyle Geopacakge is around 15MB with style and INSPIRE metadata XML embedded. The additional GeoTIFF files generated for the final map are stored in KM1000-V Raster folder and are about 430MB. However, the GeoPackage format significantly reduces the size of raster data as the KM1000FullStyleWithRaster.gpkg is only 36MB in total. Moreover, the traditional approach of sharing geospatial maps involves packing everything in a ZIP file, which requires additional ZIP software in order to unzip the file and view the content inside. The GeoPackage can be shared directly and viewed in a direct SQL interface or a web application such as DB viewer for SQLite or the NGA's application (OGC 2009).

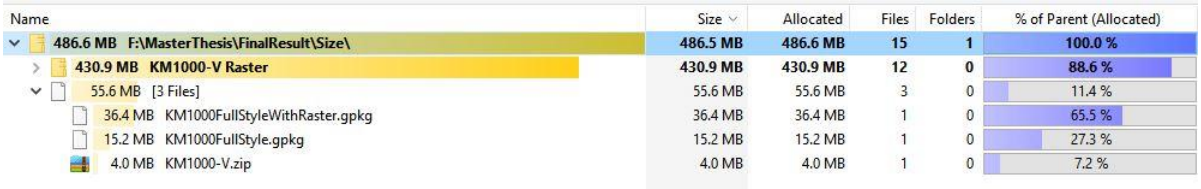



Figure 4-1. Data Size Comparison

The final map exported from QGIS is represented below (see details in Annex 3). Except for the feature labels, the map generated through the research method has successfully matched the 1:1 Million Austria Cartographic Model with national map to a large extent.

All the metadata from BEV PDF and TXT have been embedded in the GeoPackage successfully. In addition, the GeoPackage also supports storage of the INSPIRE XML metadata. Figure 4-2 presents an example of viewing the embedded metadata through DB Brower for SQLite.


Annex 3 Final Result Map exported from QGIS

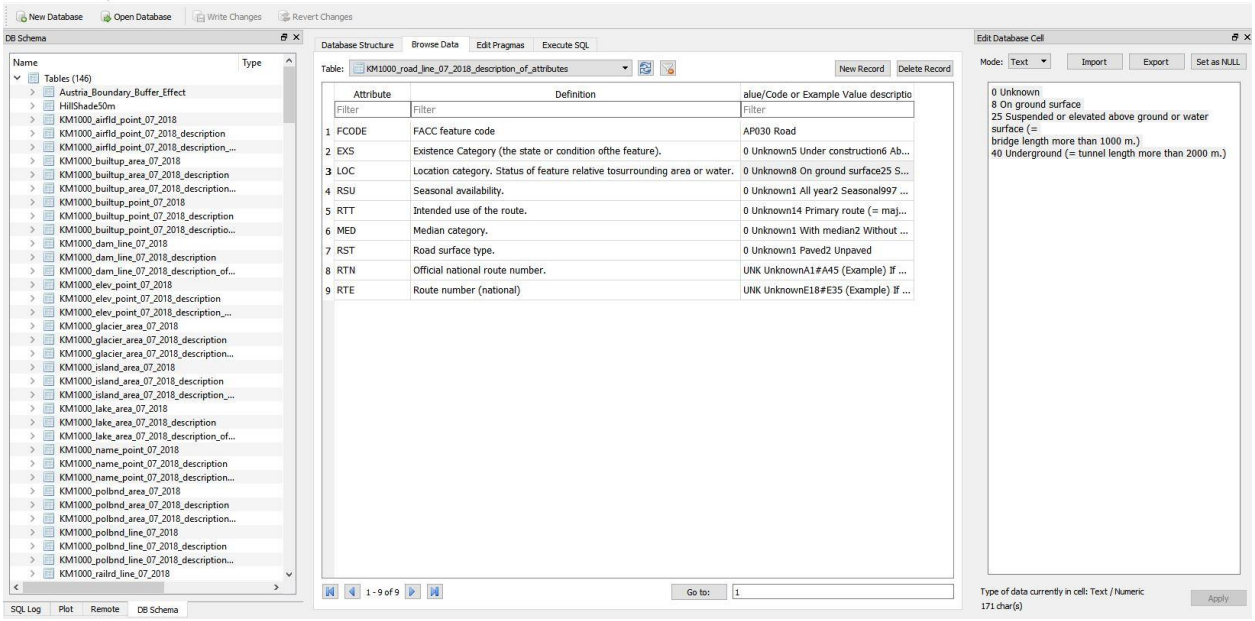



Figure 4-2. Viewing Embedded Metadata through DB Brower SQlite

## 4.2 Vector Tiles Dissemination

Regarding the dissemination of the GeoPackage, the result has demonstrated the possibility of sharing the digital map in vector tiles format through the latest GeoServer version. In addition, GeoServer also supports the distribution of the digital maps in various format such as JPEG, KML, PDF, and GeoTIFF through WMS, WFS, WCS, and WMTS.

Figure 4-3 shows the preview of the published digital map in GeoServer. One drawback of sharing the digital map through this web service is the limited support for feature label representation and special symbology created in desktop GIS software.

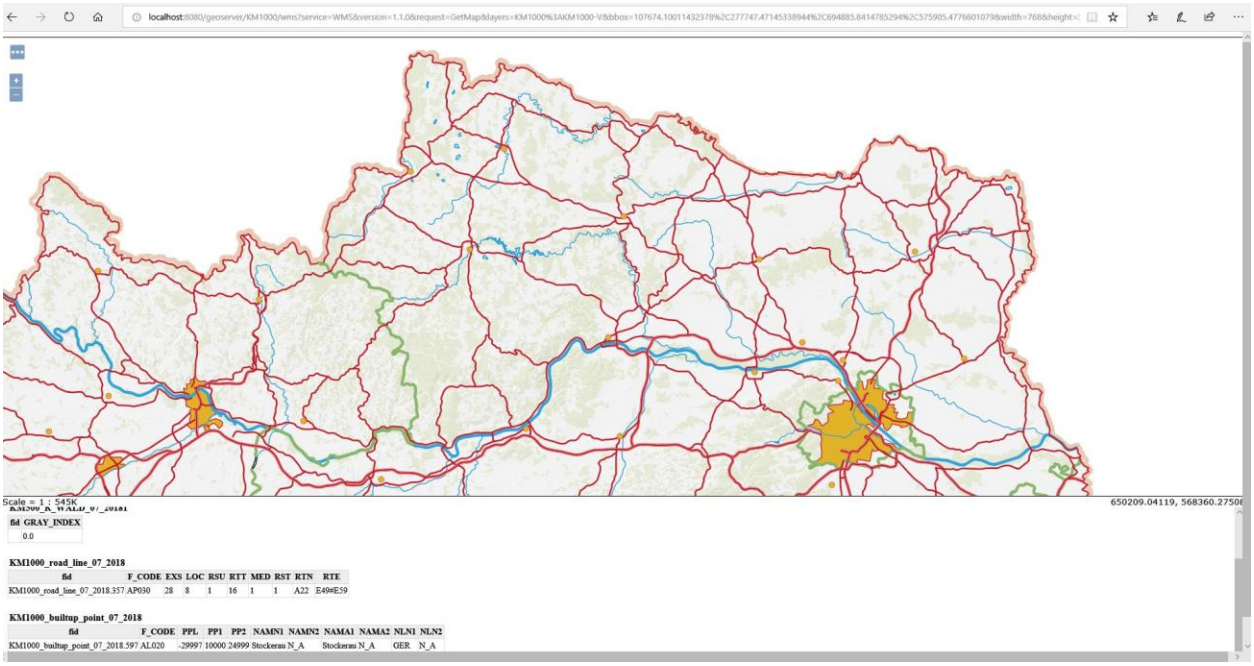



Figure 4-3. Preview of the Published Digital Maps in GeoServer

## 4.3 Python Script for Automatically Embedding Metadata

As the highlight and innovative part of this research, the created Python script has dramatically reduced the workload for the metadata embedding procedure. In order to offer a user-friendly environment, the command line scripts are provided to ease the Python dependencies installation process and a graphic user interface (GUI) is created so the user can choose the metadata document they want to embed into the GeoPackage in an easier way without modifying the source code (See Figure 4-4). The Python scripts and command line scripts have been tested in both MacOS Mojave and Windows 10 64bit through virtual machine.

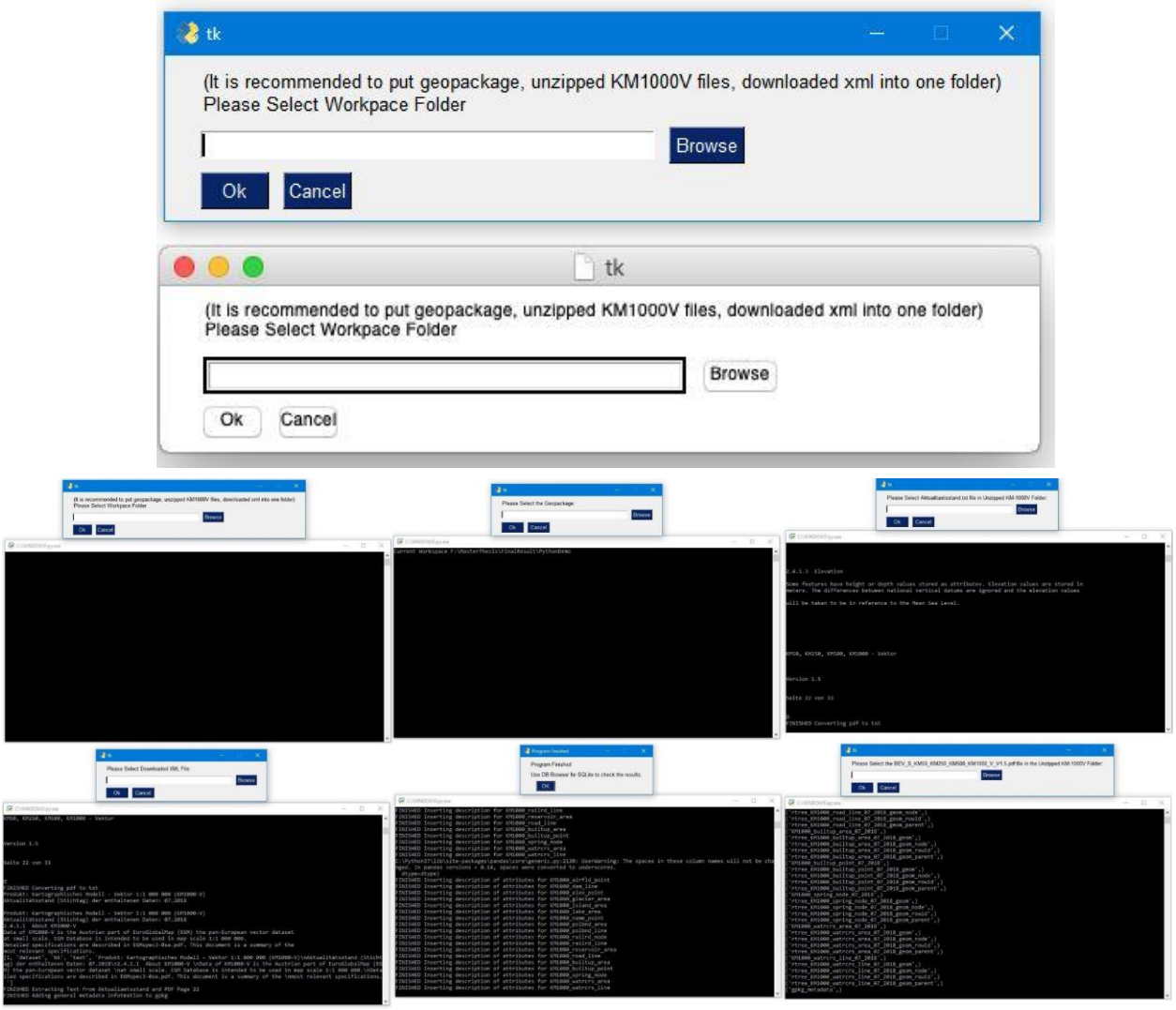



Figure 4-4. View of the GUI for the Python Scripts in MacOS and Windows 10

Although the Python script has been proved to be successful to embed metadata for GeoPackage, this Python script is designed specifically for the KM1000-V dataset from BEV. If there are certain changes of text structure or content in the BEV PDF, the Python script is subject to modify in accordance with the change in metadata. The source code of the Python script is provided in Annex 4.

# 5. Discussion and Future Work

This chapter discusses the issues that occurred during the research experiment and future work that can improve the current research method. Some of the issues have been solved and are explained in detail in the subchapters.

In order to reach the goal of this thesis and answer the research questions, a literature review was carried out and a case study for the Austrian Cartographic Model with national style from BEV was developed. In the case study, a research method was designed and a Python script was created to provide a user-friendly interface and minimize the human effort in metadata embedding procedure.

The literature review of this thesis focused on the principle, status quo, and challenge of archiving digital maps at different times and the evaluations of existing and future geospatial data formats for the preservation of digital maps. In the literature review, most of archive projects hosted by government agencies are still using SHP and GeoTIFF to store geospatial data. However, there were limited research concentrating on the disadvantage of existing geospatial data formats or proposing an alternative format for archiving digital maps because ESRI still takes the largest share in GIS industry. In addition, the research related to Geopackage format is also rare because Geopackage is a newly invented geospatial data format and it is mainly promoted by non-profit organization such as OGC. More detailed research is needed to study benefit and drawback of different geospatial data formats used in archiving digital maps. Also, it has been found by this thesis that the default metadata table provided in Geopackage Encoding Standard was not sufficient to store attribute tables and additional metadata when archiving multiple feature layers. Thus, further research is needed to study metadata extension and set new standard for metadata tables in Geopackage.

From a technical perspective, the method designed for the case study provides a completely free, open-source, and cross-platform solution for archiving and dissemination of digital maps. However, this approach relies heavily on two pieces of GIS software, which are QGIS and GeoServer. Therefore, further research is needed to examine how well proprietary GIS software such as ArcGIS and MapInfo and other open-source GIS software such as PostGIS and GRASS GIS support a Geopackage that contains metadata and style information. Moreover, as described in chapter 3, this approach uses SLD format to store style information of feature layers and GeoServer to disseminate digital maps. It is important to study the compatibility of SLD format with other web map platforms and the conversion capability between SLD format and other style formats. Furthermore, the Python script developed for this approach is exclusively designed to extract the metadata in the PDF file of BEV KM1000-V datasets. Due to the specific text structure in the PDF file, it requires modification before applying the Python script to other datasets. Additional research should look into the compatibility of using Python script to extract metadata from other PDF or TXT file.

## 5.1 Character Encoding Issue

During the experiment, it was noticed that there was an error when displaying the German umlauts characters (see Figure 5-1). This issue is relating to the character encoding type of the operating system. Since the original KM1000-V datasets are encoded in ISO 8859-1 (Latin -1), this issue does not affect the users in German speaking counties as Latin-1 may be the default character encoding for their operating system. However, if the default system encoding is UTF-8, the issue below would occur. UTF-8(ISO 10646) has become the most popular encoding method for the World Wide Web since 2009. Nowadays, over 90 percent of web pages are encoded in UTF-8 as it supports all the national languages and many non-spoken languages in the world (W3Techs, 2019 & Davis, 2012). The quick solution for this issue is to change the encoding method of the original SHPs from Latin-1 to UTF-8 before importing the SHPs to a GeoPackage. And the long-term solution should be advising the BEV to switch the encoding format from Latin-1 to UTF-8.


| NAMN1 | NAMN2 | NAMA1 | NA |
|---|---|---|---|
| M�hldorf | N_A | Muehldorf | N_A |
| Roith | N_A | Roith | N_A |
| Oberweis | N_A | Oberweis | N_A |
| Schalchham | N_A | Schalchham | N_A |
| Fuschl am See | N_A | Fuschl am See | N_A |
| Glasenbach | N_A | Glasenbach | N_A |
| Plainfeld | N_A | Plainfeld | N_A |
| Vigaun | N_A | Vigaun | N_A |
| Oberndorf | N_A | Oberndorf | N_A |
| Endach | N_A | Endach | N_A |
| B�hlerwerk | N_A | Boehlerwerk | N_A |
| Ertl | N_A | Ertl | N_A |
| Reichraming | N_A | Reichraming | N_A |

Figure 5-1. German Umlauts Encoding Issue

## 5.2 Map Labeling and Symbology Issue

As one of the most vital characters in cartographic representation, map labeling has contunued to pose challenges in the GIS industry. There are two main labeling issues relating to this research. The first one is the label abbreviation and case-sensitive issue (See Figure 5-2). For some unknown reason, the person who created the BEV national map did not use the “NAME” field in the atrribute table of KM1000_builtup_area_07_2018 to label the feature or the person changed the label text manually. Unfortunately, there is no automatic solution for this issue except for manually changing the label text to match the BEV national map.

The second labeling issue concerns the placement and dissemination of map labeling. In this case study, only part of the label information can be embedded into the GeoPackage in the SLD file such as font type and font size. In addition, there is no function in QGIS to store the label placement and export the label in a proper format that can be accessed by other GIS software or GIS web service like GeoServer. In fact, most map labeling functions nowadays are proprietary

and software-dependent. One solution for this issue can be achieved using ArcGIS desktop by converting the labels to annotation and exporting the annotation as a raster format such as PNG while the feature layer is turned off. Then, PNG can be imported in other GIS software and overlaid on other layers in the dataset. However, this solution disobeys the purpose of this research which promotes a free, cross-platform, and open-source environment for the preservation and dissemination of digital maps. Thus, future work should focus on the proper method to export and share the label placement in QGIS.


BEV National Map Final Map of this research

Figure 5-2. Map Labelling and Source Data Issue

Although similar or equal symbols can be created in QGIS, a web map service like GeoServer has its limitations to display the symbol in a complex structure. For example, two of the road symbols do not display in the final map on GeoServer. Future work should look into the performance and capability of SLD and alternative style formats like CSS. In addition, further research can explore the style editor function from alternative web map services such as Mapbox and Mapstore2.

Second, the latest GeoServer version has a case-sensitive issue to recognize the symbology from the uploaded SLD file. This issue did not occur in earlier version such as GeoServer 2.14.1. The solution for this issue is to check error code and change the corresponding lower case letters in the field name of the attribute table to upper case.

## 5.3 Geodata Quality Issue

During the orginal dataset review stage and editing style stage, geodata quality issues in the KM1000-V dataset have occurred. First, only 9 out 18 layer from the original dataset were used to make the 1:1 million BEV nationl map. Second, features data are missing from certain layer such as the KM1000_builtup_area_07_2018. Using Figure 5-2 as an example, the city boundary of Sankt Pölten is missing from the original dataset. Moreover, features in some layers such as the road line, watercourse line, and watercourse area are not displayed in the BEV national map. The person creating the BEV national map had filtered out the features in those layers, which was not based on any feature characteristics stored in the attribute table the layer. This issue is solved in the KM1000FullStyleWithRaster.gpkg by manually removing the redundant features from layers. Third, the topological issue of the KM1000_polbnd_line_07_2018 layer had prevented the buffer process during the style editing stage. Thus, the cartographic representation

of the Austria border was generated by the free country boundaries data from the Databases of Global Administrative Area (DADM), which results in a gap at certain edges of the border. For the future, it is recommended that BEV personnel work to improve the source data quality.

# 6. Conclusion

Considering the limitations of current geospatial data formats to be suitable for long-term digital data preservation, and in light of the scenario illustrated in the Chapter Four, in this thesis we designed a case study approach to explore an alternative data format, GeoPackage, for the preservation and dissemination of digital maps in the future.

The research questions addressed in the first chapter can now be answered with the positive results attained through the research.

1. Is it possible to embed relevant metadata for future usage in digital maps using GeoPackage?

This case study, archiving the 1:1 million Austria Cartographic Model with national map style, demonstrates the possibility of using GeoPackage to embed relevant metadata from different data formats that the digital maps requires, such as general information for the whole dataset, layer description, description of attribute table, symbology, and reference system information. In addition, considering the widespread availability of data from BEV in the Europe Union, the capability of embedding INSPIRE XML metadata in GeoPackage was tested as well. The results have indicated several advantages of archiving digital maps with GeoPackage compared to the most commonly used data format like ESRI shapefile and GeoTIFF. Unlike shapefile and GeoTIFF format, GeoPackage has the ability to store all related data in one file. Moreover, it can store both vector and raster data, and the size of raster data significantly decreased when it was stored in GeoPackage.

2. Is it possible to embed styling from GeoPackage when distributing digital maps that use vector-tile format?

Through the last three stages in this research approach, the goal of distributing digital maps through vector-tile format has been achieved by uploading the embedded SLD style from GeoPackage to Geoserver. However, due to the technical limitations of the web map service, the style from GeoPackage cannot be fully displayed through GeoServer. Future studies are needed to explore an alternative approach to display style or improve the current functionally of this web map service.

- Is it possible to create a Python script that partially automate the procedures of embedding relevant information into GeoPackage?

In the case study, a Python scripts with GUI interface has been developed to automatically embed the related information from the BEV PDF, BEV TXT file, and INSPISE XML metadata file. The Python script has significantly reduced the workload of embedding metadata into a GeoPackage. In addition, the script has been tested in both Windows 10 and MacOS. Nevertheless,

the script is designed exclusively for the case study. Thus, any structural change of in the PDF in the future will require modification of the source code. Additional work is needed to improve the ability to extract specific parts of text from a PDF and to generate tables for it. Besides, it is also used to develop a script that can embed SLD files into a GeoPackage.

In sum, this thesis proposes an innovative approach to embed relevant metadata for future usage in digital maps using GeoPackage and to embed styling from GeoPackage when distributing digital maps that use vector-tile format. It also shows the possibility of using Python scripts to automate the procedure of embedding relevant information into GeoPackage.

# 7. List of References


Bleakly, D. R. (2002). Long-Term Spatial Data Preservation and Archiving: What Are the Issues? doi:10.2172/793225

Lauriault T.P., Pulsifer P.L., Taylor D.F. (2011). The Preservation and Archiving of Geospatial Digital Data: Challenges and Opportunities for Cartographers. In: Jobst M. (eds) *Preservation in Digital Cartography. Springer, Berlin, Heidelberg*

Ružicka, J. (2016). Comparing speed of Web Map Service with GeoServer on ESRI Shapefile and PostGIS. *Geoinformatics FCE CTU*,15(1), 3. doi:10.14311/gi.15.1.1

Bernhard, K. (2013, January 30). Austria: Leading the Open Government Implementation Model. Retrieved from https://www.opengovpartnership.org/stories/austria-leading-open-government-implementation-model

Alban, S. (2015, March 2). Independent Report Highlights ESRI as Leader in Global GIS Market. Retrieved from https://www.ESRI.com/ESRI-news/releases/15-1qtr/independent-report-highlights-ESRI-as-leader-in-global-gis-market

Theobald, D. M. (2001). Understanding Topology and Shapefiles. Retrieved from https://www.ESRI.com/news/arcuser/0401/topo.html

Cepicky, J., & OpenGeoLabs. (2017). Switch from Shapefile. Retrieved from http://switchfromshapefile.org/

Antoniou, V., Morley, J., & Haklay, M. (2009). Tiled Vectors: A Method for Vector Transmission over the Web. *W2GIS*.

Veenendaal, B., Brovelli, M. A., & Li, S. (2017). Review of Web Mapping: Eras, Trends and Directions. *ISPRS International Journal of Geo-Information, 6(10), 317. MDPI AG*. Retrieved from http://dx.doi.org/10.3390/ijgi6100317

H. Rashidan, M & Musliman, Ivin & A. Rahman, A. (2016). GEOPACKAGE DATA FORMAT FOR COLLABORATIVEMAPPING OF GEOSPATIAL DATAIN LIMITED NETWORKENVIRONMENTS. ISPRS - *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences.* XLII-4/W1. 15-21. 10.5194/isprs-archives-XLII-4-W1-15-2016.

Sanner, Michel. (1998). Python: A Programming Language for Integration and Development. 10.1016/S1093-3263(99)99999-0.

Janée, Greg. (2009). Preserving Geospatial Data: The National Geospatial Digital Archive's Approach. 1509.

Jobst, Markus & Gartner, Georg (2011). Structural Aspect for the Digital Cartographic Heritage. In: Jobst M. (eds) *Preservation in Digital Cartography. Springer, Berlin, Heidelberg*

Federal Office of Topography swisstopo(2016). Handbook: Archiving of federal geodata. https://www.swisstopo.admin.ch/en/knowledge-facts/geoinformation/landschaftsgedaechtnis/langzeitaufbewahrung/archivierung-geodaten.html

Clark, John H (2016). The Long-Term Preservation of Digital Historical Geospatial Data: A Review of Issues and Methods, *Journal of Map & Geography Libraries*, 12:2, 187-201, DOI: 10.1080/15420353.2016.1185497

Shaon, Arif & Naumann, Kai & Kirstein, Michael & Roensdorf, Carsten & Mason, Paul & Bos, Marguérite & Gerber, Urs & Woolf, Andrew & Samuelsson, Göran. (2011). Long-term sustainability of spatial data infrastructures: a metadata framework and principles of geo-archiving.

Sandner, Peter (2011). The Archiving of Digital Geographical Information. In: Jobst M. (eds) *Preservation in Digital Cartography. Springer, Berlin, Heidelberg*

Cartwright, William (2011) From Plan Press to Button Push: The Development of Technology for Cartographic Archiving and Access. *In: Jobst M. (eds) Preservation in Digital Cartography. Springer, Berlin, Heidelberg*

Zaslavsky, I. (2001). Archiving spatial data: Research issues. *La Jolla, CA: San Diego Supercomputer Center.*

Erwin, Tracey & Sweetkind-Singer, Julie (2009) The National Geospatial Digital Archive: A Collaborative Project to Archive Geospatial Data, *Journal of Map & Geography Libraries*, 6:1, 6-25, DOI: 10.1080/15420350903432440

Locher, Anita-E. & Termens, Miquel(2012). Exploring alternatives for geodata preservation, *In 7ª Conferencia Ibérica de Sistemas y Tecnologías de Información*. CISTI 2012, Madrid

Morris, Steven P., and James Tuttle (2007). Curation and Preservation of Complex Data: The North Carolina Geospatial Data Archiving Project. North Carolina State University Libraries.

Pons, X., & Masó-Pau, J. (2016). A comprehensive open package format for preservation and distribution of geospatial data and metadata. *Computers & Geosciences*, 97, *89-97.*

ESRI(1998). ESRI Shapefile Technical Description. An ESRI White Paper. https://www.esri.com/library/whitepapers/pdfs/shapefile.pdf

North Carolina Center for Geographic Information and Analysis & North Carolina Department of Cultural Resources (2011). Geospatial Multistate Archive and Preservation Partnership Final Report

Guptill, Stephen (1999) Metadata and Data Catalogues. *Geographical Information Systems/Longley, PA et al.* pp 677-692

FGDC (2011). Geospatial Metadata Fact Sheet. Retriever from https://www.fgdc.gov/resources/factsheets/documents/GeospatialMetadata-July2011.pdf

ESRI (2019). FAQ: What is the difference between a shapefile and a layer file?. Retrieved from https://support.esri.com/en/technical-article/000011516

Ritter, Niles (1995). GeoTIFF Fomat Specification, retriever from http://mac.mf3x3.com/GIS/GEOTIFF/geotiff_spec.pdf

Mahammad, Sazid & Ramakrishnan, R (2003). GeoTIFF – A Standard Image File Format for GIS Application

Bogossian, C.H., Ferreira, K.R., Monteiro, A.M., & Vinhas, L. (2014). A Hybrid Architecture for Mobile Geographical Data Acquisition and Validation Systems. *GeoInfo*.

Open Geospatial Consortium (2018) OGC GeoPackage Encoding Standard – With Corrigendum. Retrieved from https://www.opengeospatial.org/standards/GeoPackage

Li L, Hu W, Zhu H, Li Y, Zhang H (2017) Tiled vector data model for the geographical features of symbolized maps. *PLoS ONE* 12(5): e0176387. https://doi.org/10.1371/journal.pone.0176387

Ordnance Survey (2018) OS Open Zoomstack Vector Tiles. Retriever from https://www.ordnancesurvey.co.uk/business-and-government/help-and-support/products/os-open-zoomstack.html

Ordnance Survey (2019) OS Open Zoomstack Technical Specification. Retriever from https://www.ordnancesurvey.co.uk/business-and-government/help-and-support/products/os-open-zoomstack.html

Altaweel, M. (2017). The Use of Python in GIS. Retrieved from https://www.gislounge.com/use-python-gis/

ESRI (2018). What is Python? Retrieved from http://desktop.arcgis.com/en/arcmap/latest/analyze/python/what-is-python-.htm

INSPIRE MIG (2007). Article 5(1) of INSPIRE Directive 2007/2/EC, Retrieved from https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32007L0002&rid=1

INSPIRE MIG (2017). Technical Guidelines for implementing dataset and service metadata based on ISO/TS 19139:2007

Open Geospatial Consortium (2007). Styled Layer Descriptor Profile of the Web Map Service Implementation Specification. Retrieved from http://www.opengeospatial.org/standards/sld#overview

Open Geospatial Consortium (2019) Getting Started with GeoPakcage. Retrieved from http://www.GeoPackage.org/guidance/getting-started.html

Davis, Mark (2012). Unicode over 60 percent of the web. Retrieved from https://googleblog.blogspot.com/2012/02/unicode-over-60-percent-of-web.html.

W3TECHs.com (2019). Usage Survey of Character Encodings broken down by Ranking. Retriev ed from https://w3techs.com/technologies/cross/character_encoding/ranking

# Appendix

## Annex 1 – Command Line Scripts for Windows OS

BatchInstall_windows.bat

```
@echo off
rem this is a batch file to install all required python modules for the project.

echo Please drag your python folder here(example C:\python37)

set /p pypath= "Python Folder Location:"

pushd %pypath%\Scripts

python.exe -m pip install --upgrade pip

pip install PySimpleGUI
pip install tabula
pip install distro
pip install urllib3
pip install pdfminer.six
pip install pandas
pip install DBManager
pip install datefinder

pause
```

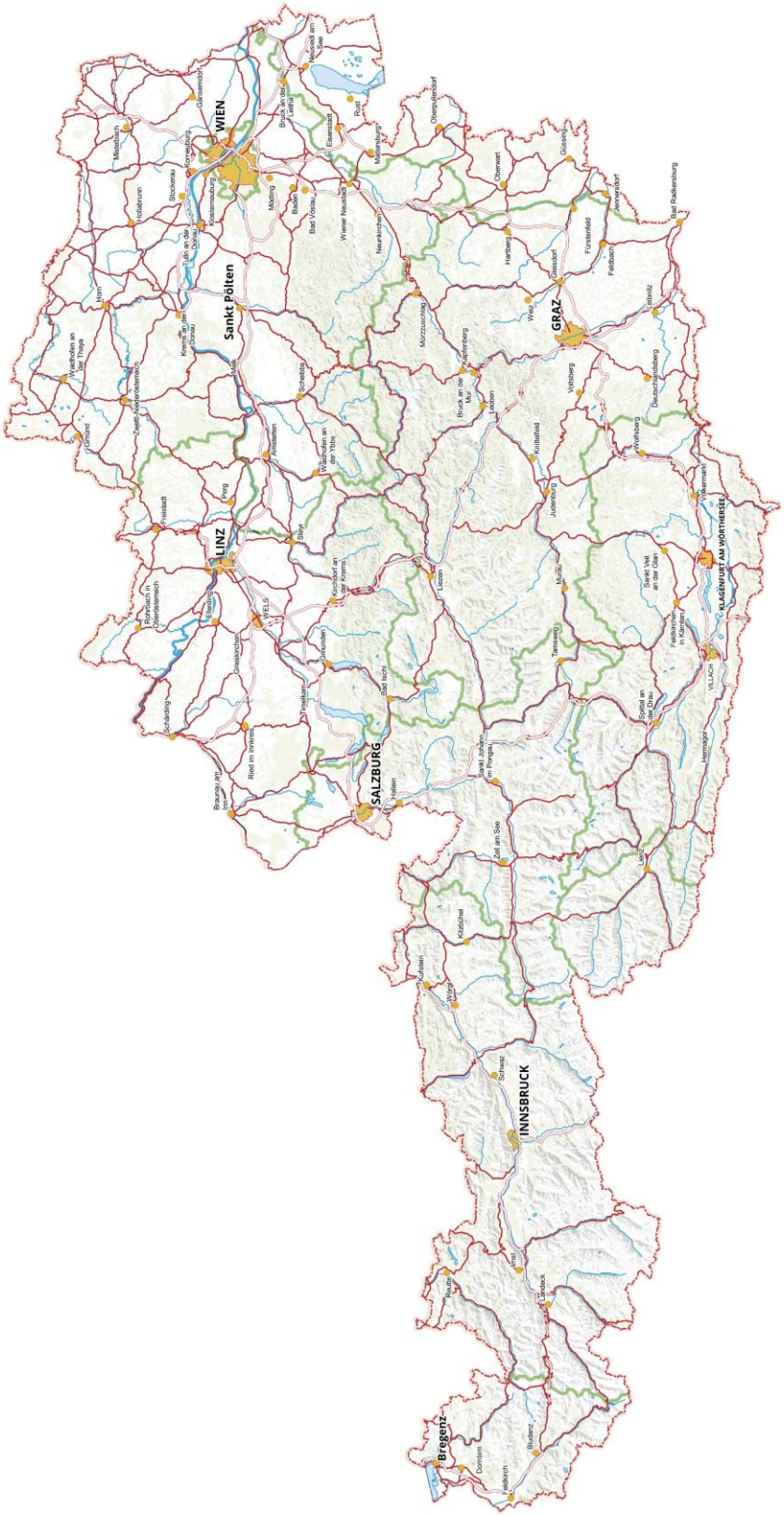
## Annex 2 – Command Line Scripts for MacOS

BatchInstall_MacOS.command

```
#! /bin/bash
%echo This batch file will install required python modules for the project.

pip3 install --upgrade pip
Pip3 install tabula-py
Pip3 install urllib3
pip3 install distro
pip3 install pdfminer.six
pip3 install pandas
pip3 install DBManager
pip3 install datefinder
pip3 install tkinter
pip3 install PySimpleGUI
```

## Annex 3 – Final Result Map exported from QGIS



## Annex 3 – Source Code of the Python Script

```
##Master Thesis:Archiving Digital Maps with GeoPackage and Vector-tile Dissemination
##Author: Yunnan Chen
##tabula-py requires java JDK, pandas, urllib3, distro

import PySimpleGUI as sg
import re, sys, os, datefinder
from shutil import copyfile
import sqlite3,DBManager
import pandas as pd
from pandas.io.sql import to_sql, read_sql
from pdfminer.pdfinterp import PDFResourceManager, PDFPageInterpreter
from pdfminer.converter import TextConverter
from pdfminer.layout import LAParams
from pdfminer.pdfpage import PDFPage
from io import StringIO
import tabula
from tabula import read_pdf
import csv
from xml.etree import ElementTree
import xml.dom.minidom
import datetime

##Output Text Location
##cwd = os.getcwd()
##print ("Current Workspace",cwd)
CurrentWorkspace = sg.PopupGetFolder('(It is recommended to put GeoPackage, unzipped KM 1000V files, downloaded xml into one folder)\n'+'Please Select Workpace Folder')

if CurrentWorkspace != "":
    sg.Popup('The Current Workspace is ', CurrentWorkspace)
    os.chdir(CurrentWorkspace)
    cwd = os.getcwd()
    print ("Current Workspace",cwd)
else:
    sg.Popup('The Current Workspace is not defined!','Program Stopped')
    sys.exit("Error")

##Path Location of GeoPackage
GeoPackage = sg.PopupGetFile('Please Select the GeoPackage:')
if GeoPackage != "":
    conn = sqlite3.connect(GeoPackage)
    sg.Popup(GeoPackage, 'Database Connection Successful')
else:
    sg.Popup(GeoPackage, 'Database Connection Failed','Program Stopped')
    sys.exit("Error")

print ("Database Connection Successful")

cursor = conn.cursor()

##Create gpkg_metadata table
cursor.execute('DROP TABLE IF EXISTS gpkg_metadata')
cursor.execute('''''CREATE TABLE gpkg_metadata (
  id INTEGER CONSTRAINT m_pk PRIMARY KEY ASC NOT NULL,
  md_scope TEXT NOT NULL DEFAULT 'dataset',
  md_standard_uri TEXT NOT NULL,
  mime_type TEXT NOT NULL DEFAULT 'text/xml',
```

```
  metadata TEXT NOT NULL DEFAULT ''
);''')
conn.commit()

##List all tables in the database
cursor.execute("SELECT name FROM sqlite_master WHERE type='table';")
tables = cursor.fetchall()
for table in tables:
    print (table)

##Create gpkg_metadata_reference table
cursor.execute('DROP TABLE IF EXISTS gpkg_metadata_reference')
cursor.execute('''''CREATE TABLE gpkg_metadata_reference (
  reference_scope TEXT NOT NULL,
  table_name TEXT,
  column_name TEXT,
  row_id_value INTEGER,
  timestamp DATETIME NOT NULL DEFAULT (strftime('%Y-%m-%dT%H:%M:%fZ','now')),
  md_file_id INTEGER NOT NULL,
  md_parent_id INTEGER,
  CONSTRAINT crmr_mfi_fk FOREIGN KEY (md_file_id) REFERENCES gpkg_metadata(id),
  CONSTRAINT crmr_mpi_fk FOREIGN KEY (md_parent_id) REFERENCES gpkg_metadata(id)
);''')
conn.commit()

##Export BEV PDF as Text
BEVPDF = sg.PopupGetFile('''''Please Select the BEV_S_KM50_KM250_KM500_KM1000_V_V1.5.pdf file in the Unzipped KM-1000V Folder:''')
if BEVPDF != "":
    sg.Popup(BEVPDF, 'Import Successful')
else:
    sg.Popup(BEVPDF, 'Import Failed','Program Stopped')
    sys.exit("Error")

##Convert PDF to Text
##Source Code: http://stanford.edu/~mgorkove/cgi-bin/rpython_tutorials/Using%20Python%20to%20Convert%20PDFs%20to%20Text%20Files.php
def convert(fname, pages=None):
    if not pages:
        pagenums = set()
    else:
        pagenums = set(pages)

    codec = 'utf-8'
    output = StringIO()
    manager = PDFResourceManager()
            converter = TextConverter(manager, output, laparams=LAParams())
            interpreter = PDFPageInterpreter(manager, converter)

            infile = open(fname, 'rb')
            for page in PDFPage.get_pages(infile, pagenums):
                interpreter.process_page(page)
            infile.close()
            converter.close()
            text = output.getvalue()
            output.close
            return text

       output = convert(BEVPDF, pages=[21])
       print (output)

```

```
        file = open("PDFPage22.txt", "w", encoding='utf-8')
        file.write(output)
        file.close()
        print ("FINISHED Converting pdf to txt")

        ##Path Location of Aktualitaetsstand
        LatestInfo = sg.PopupGetFile('Please Select Aktualitaetsstand.txt file in Unzipped KM-1000V Folder:')
        if LatestInfo !="":
            sg.Popup(LatestInfo, 'Import Successful')
        else:
            sg.Popup(LatestInfo, 'Import Failed','Program Stopped')
            sys.exit("Error")

        lines = []

        file = open(LatestInfo, 'rt',encoding="latin-1")

        for line in file:
            lines.append(line)

        ReleaseDate = lines[3] + lines[5]
        print (ReleaseDate)


        ##Extract General Structure Information of KM-1000
        ExtractGeneralStructure = "PDFPage22.txt"
        file = open(ExtractGeneralStructure, 'r',encoding='utf-8')
        content = file.read()
        GeneralStructurePattern = r'2.4.1.1  About KM1000-V (.*\n.*\n.*\n.*\n.*) .*'
        test = re.search(GeneralStructurePattern, content, re.MULTILINE)
        GeneralStructure = str(test.group())
        GeneralInformation = ReleaseDate + GeneralStructure
        print (GeneralInformation)

        ##Append general metadata infotmation to gpkg
        metadata = [1,'dataset','NA', 'text', GeneralInformation]
        print (metadata)
        cursor.execute('insert into gpkg_metadata values (?,?,?,?,?)', metadata)
        conn.commit()
        print ('''''FINISHED Extracting Text from Aktualiaetsstand and PDF Page 22
        FINISHED Adding general metadata infotmation to gpkg''')

        file.close()
        if os.path.exists("PDFPage22.txt"):
          os.remove("PDFPage22.txt")
        else:
          print("The file does not exist")

        ##Path Location of the XML File
        polbnd_line_xml = sg.PopupGetFile('Please Select Downloaded XML File:')
        if LatestInfo !="":
            sg.Popup(polbnd_line_xml, 'Import Successful')
        else:
            sg.Popup(polbnd_line_xml, 'Import Failed','Program Stopped')
            sys.exit("Error")

        ##Append KM1000_polbnd_line XML to gpkg
        polbnd_line_xml_copy = "polbnd_line_xml_copy.xml"
        copyfile(polbnd_line_xml, polbnd_line_xml_copy)

```

```
        xmlformatpreserved = xml.dom.minidom.parse(polbnd_line_xml_copy)
        xmlformatpreserved = xmlformatpreserved.toprettyxml()
        xmlformatpreserved = str(xmlformatpreserved)

        xmlLink = "http://geometadatensuche.inspire.gv.at/metadatensuche/srv/eng/catalog.search#/metadata/d2b8d67f-737c-4d49-b220-ca0ef422197d"
        metadata = [2,'feature',xmlLink, 'xml', xmlformatpreserved]
        cursor.execute('insert into gpkg_metadata values (?,?,?,?,?)', metadata)
        conn.commit()
        print ("Finishing inserting xml to GeoPackage")

        timestamp = datetime.datetime.now()
        metadata_reference = ['table','KM1000_POLBND_LINE_07_2018', '', '',
                              timestamp, 2, 1]
        cursor.execute('insert into gpkg_metadata_reference values (?,?,?,?,?,?,?)', metadata_reference)
        conn.commit()

        def replace_in_config(old, new):
            with open(polbnd_line_xml_copy, 'r') as f:
                text = f.read()

            with open(polbnd_line_xml_copy, 'w') as f:
                f.write(text.replace(old, new))

        replace_in_config('gco:', '')
        replace_in_config('gmd:', '')


        tree = ElementTree.parse(polbnd_line_xml_copy)
        root = tree.getroot()

        extractxml = ""

        for child in root.iter():
            tagstring = str(child.tag)
            textstring = str(child.text)
            extractxml += tagstring + textstring

        extractxml = str(extractxml)

        metadata = [3,'feature',xmlLink, 'text', extractxml]
        cursor.execute('insert into gpkg_metadata values (?,?,?,?,?)', metadata)
        conn.commit()

        timestamp = datetime.datetime.now()
        metadata_reference = ['table','KM1000_POLBND_LINE_07_2018', '', '',
                              timestamp, 3, 1]
        cursor.execute('insert into gpkg_metadata_reference values (?,?,?,?,?,?,?)', metadata_reference)
        conn.commit()

        print ("Finishing inserting xml to GeoPackage")
        if os.path.exists("polbnd_line_xml_copy.xml"):
            os.remove("polbnd_line_xml_copy.xml")
        else:
            print("The file does not exist")

        ##Convert tables in PDF to CSV
        tabula.convert_into(BEVPDF,"DataFormatFileTable.csv",encoding='utf-8',multiple_tables= True,output_format='data_format',pages="22")
```

```
232.        tabula.convert_into(BEVPDF,"tablepage23.csv",encoding='utf-
    8',guess = False, multiple_tables= True,output_format='data_format',pages="23")
233.        tabula.convert_into(BEVPDF,"tablepage24.csv",encoding='utf-
    8',guess = True, lattice = True, multiple_tables= True,output_format='data_format',page
    s="24")
234.        tabula.convert_into(BEVPDF,"tablepage25.csv",encoding='utf-
    8',guess = True, lattice = True, multiple_tables= True,output_format='data_format',page
    s="25")
235.        tabula.convert_into(BEVPDF,"tablepage2627.csv",encoding='utf-
    8',guess = True, lattice = True, multiple_tables= True,output_format='data_format',page
    s="26,27")
236.        tabula.convert_into(BEVPDF,"tablepage28.csv",encoding='utf-
    8',guess = True, lattice = True, multiple_tables= True,output_format='data_format',page
    s="28")
237.        tabula.convert_into(BEVPDF,"tablepage2930.csv",encoding='utf-
    8',guess = True, lattice = True, multiple_tables= True,output_format='data_format',page
    s="29,30")
238.        tabula.convert_into(BEVPDF,"tablepage31.csv",encoding='utf-
    8',guess = True, lattice = True, multiple_tables= True,output_format='data_format',page
    s="31")
239.        tabula.convert_into(BEVPDF,"tablepage32.csv",encoding='utf-
    8',guess = True, lattice = True, multiple_tables= True,output_format='data_format',page
    s="32")
240.        tabula.convert_into(BEVPDF,"tablepage33.csv",encoding='utf-
    8',guess = True, lattice = True, multiple_tables= True,output_format='data_format',page
    s="33")
241.        print ("FINISHED EXTRACT Table from PDF")
242.
243.        ##Insert Description for gpkg_contents
244.        LayerDescription = pd.read_csv("DataFormatFileTable.csv", usecols = ['Descriptio
    n'],nrows = 18)
245.        LayerDescription = pd.DataFrame(LayerDescription)
246.        LayerDescription = LayerDescription['Description'].tolist()
247.        print (LayerDescription)
248.
249.        cursor.execute('SELECT srs_id FROM gpkg_contents ORDER BY srs_id')
250.        srs_id = [row[0] for row in cursor]
251.        for a, b in zip(LayerDescription, srs_id):
252.            cursor.execute('UPDATE gpkg_contents SET description = ? WHERE srs_id = ?',
    [a, b])
253.        conn.commit()
254.        print ("FINISHED Inserting description for gpkg_contents")
255.        if os.path.exists("DataFormatFileTable.csv"):
256.            os.remove("DataFormatFileTable.csv")
257.        else:
258.            print("The file does not exist")
259.
260.        ##Manually add description of layer to gpkg for KM1000_airfld_point
261.        FeatureClassName = "KM1000_airfld_point"
262.        Definition = "A defined area used for landing, take-
    off, and movement of aircraft including associated buildings and facilities"
263.        EGM_Feature_Class = "AirfldP"
264.        FeatureType = "Point"
265.        PrimitiveType = "Isolated node"
266.        PortrayalCriteria = "All airports having regular passenger traffic"
267.        attributeTable = "KM1000_airfld_point_07_2018_description_of_attributes"
268.        table = [1,FeatureClassName, Definition, EGM_Feature_Class, FeatureType, Primiti
    veType, PortrayalCriteria]
269.        cursor.execute('DROP TABLE IF EXISTS KM1000_airfld_point_07_2018_description')
270.        cursor.execute('''''CREATE TABLE KM1000_airfld_point_07_2018_description (
271.                id INTEGER CONSTRAINT m_pk PRIMARY KEY ASC NOT NULL,
```

```
                featureClassName TEXT NOT NULL,
                definition TEXT NOT NULL,
                EGMFeatureClass TEXT NOT NULL,
                featureType TEXT NOT NULL,
                primitiveType TEXT NOT NULL,
                portrayalCriteria TEXT NOT NULL
                );''')
        cursor.execute('INSERT INTO KM1000_airfld_point_07_2018_description VALUES (?,?,?,?,?,?,?)', table)
        conn.commit()
        print ("FINISHED Inserting description for KM1000_airfld_point")

        ##Manually add description of layer to gpkg for KM1000_dam_line
        FeatureClassName = "KM1000_dam_line"
        Definition = "A permanent barrier across a watercourse used to impound water or to control its flow."
        EGM_Feature_Class = "DamL"
        FeatureType = "Line"
        PrimitiveType = "Edge"
        PortrayalCriteria = "Dams with remarkable national meaning or longer than 2000 meters."
        attributeTable = "KM1000_dam_line_07_2018_description_of_attributes"
        table = [1,FeatureClassName, Definition, EGM_Feature_Class, FeatureType, PrimitiveType, PortrayalCriteria]
        cursor.execute('DROP TABLE IF EXISTS KM1000_dam_line_07_2018_description')
        cursor.execute('''''CREATE TABLE KM1000_dam_line_07_2018_description (
                id INTEGER CONSTRAINT m_pk PRIMARY KEY ASC NOT NULL,
                featureClassName TEXT NOT NULL,
                definition TEXT NOT NULL,
                EGMFeatureClass TEXT NOT NULL,
                featureType TEXT NOT NULL,
                primitiveType TEXT NOT NULL,
                portrayalCriteria TEXT NOT NULL
                );''')
        cursor.execute('INSERT INTO KM1000_dam_line_07_2018_description VALUES (?,?,?,?,?,?,?)', table)
        conn.commit()
        print ("FINISHED Inserting description for KM1000_dam_line")

        ##Manually add description of layer to gpkg for KM1000_elev_point
        FeatureClassName = "KM1000_elev_point"
        Definition = "A designated location with an elevation value relative to a vertical datum."
        EGM_Feature_Class = "ElevP"
        FeatureType = "Point"
        PrimitiveType = "Isolated node"
        PortrayalCriteria = "1 - 30 remarkable height points for each country. At least the highest point of the country."
        attributeTable = "KM1000_elev_point_07_2018_description_of_attributes"
        table = [1,FeatureClassName, Definition, EGM_Feature_Class, FeatureType, PrimitiveType, PortrayalCriteria]
        cursor.execute('DROP TABLE IF EXISTS KM1000_elev_point_07_2018_description')
        cursor.execute('''''CREATE TABLE KM1000_elev_point_07_2018_description (
                id INTEGER CONSTRAINT m_pk PRIMARY KEY ASC NOT NULL,
                featureClassName TEXT NOT NULL,
                definition TEXT NOT NULL,
                EGMFeatureClass TEXT NOT NULL,
                featureType TEXT NOT NULL,
                primitiveType TEXT NOT NULL,
                portrayalCriteria TEXT NOT NULL
                );''')
```

```
        cursor.execute('INSERT INTO KM1000_elev_point_07_2018_description VALUES (?,?,?,?,?,?,?)', table)
        conn.commit()
        print ("FINISHED Inserting description for KM1000_elev_point")

        ##Manually add description of layer to gpkg for KM1000_glacier_area
        FeatureClassName = "KM1000_glacier_area"
        Definition = "A large mass of snow and ice moving slowly down a slope or valley from above the snowline."
        EGM_Feature_Class = "LandiceA"
        FeatureType = "Area"
        PrimitiveType = "Face"
        PortrayalCriteria = "Glaciers larger than 3 km²."
        attributeTable = "KM1000_glacier_area_07_2018_description_of_attributese"
        table = [1,FeatureClassName, Definition, EGM_Feature_Class, FeatureType, PrimitiveType, PortrayalCriteria]
        cursor.execute('DROP TABLE IF EXISTS KM1000_glacier_area_07_2018_description')
        cursor.execute('''''CREATE TABLE KM1000_glacier_area_07_2018_description (
                id INTEGER CONSTRAINT m_pk PRIMARY KEY ASC NOT NULL,
                featureClassName TEXT NOT NULL,
                definition TEXT NOT NULL,
                EGMFeatureClass TEXT NOT NULL,
                featureType TEXT NOT NULL,
                primitiveType TEXT NOT NULL,
                portrayalCriteria TEXT NOT NULL
                );''')
        cursor.execute('INSERT INTO KM1000_glacier_area_07_2018_description VALUES (?,?,?,?,?,?,?)', table)
        conn.commit()
        print ("FINISHED Inserting description for KM1000_glacier_area")

        ##Manually add description of layer to gpkg for KM1000_island_area
        FeatureClassName = "KM1000_island_area"
        Definition = "A land mass smaller than a continent and surrounded by water"
        EGM_Feature_Class = "IslandA"
        FeatureType = "Area"
        PrimitiveType = "Face"
        PortrayalCriteria = "Islands larger than 3 km². Smaller islands in water area can be portrayed if considered as landmark because containing an important settlement, etc."
        attributeTable = "KM1000_island_area_07_2018_description_of_attributes"
        table = [1,FeatureClassName, Definition, EGM_Feature_Class, FeatureType, PrimitiveType, PortrayalCriteria]
        cursor.execute('DROP TABLE IF EXISTS KM1000_island_area_07_2018_description')
        cursor.execute('''''CREATE TABLE KM1000_island_area_07_2018_description (
                id INTEGER CONSTRAINT m_pk PRIMARY KEY ASC NOT NULL,
                featureClassName TEXT NOT NULL,
                definition TEXT NOT NULL,
                EGMFeatureClass TEXT NOT NULL,
                featureType TEXT NOT NULL,
                primitiveType TEXT NOT NULL,
                portrayalCriteria TEXT NOT NULL
                );''')
        cursor.execute('INSERT INTO KM1000_island_area_07_2018_description VALUES (?,?,?,?,?,?,?)', table)
        conn.commit()
        print ("FINISHED Inserting description for KM1000_island_area")

        ##Manually add description of layer to gpkg for KM1000_lake_area
        FeatureClassName = "KM1000_lake_area"
        Definition = "A body of water surrounded by land."
```

```
        EGM_Feature_Class = "LakeresA"
        FeatureType = "Area"
        PrimitiveType = "Face"
        PortrayalCriteria = "Lakes larger than 0.5 km². Lakes being part of the water ne
    twork have to be topologically connected to watercourses."
        attributeTable = "KM1000_lake_area_07_2018_description_of_attributes"
        table = [1,FeatureClassName, Definition, EGM_Feature_Class, FeatureType, Primiti
    veType, PortrayalCriteria]
        cursor.execute('DROP TABLE IF EXISTS KM1000_lake_area_07_2018_description')
        cursor.execute('''''CREATE TABLE KM1000_lake_area_07_2018_description (
                id INTEGER CONSTRAINT m_pk PRIMARY KEY ASC NOT NULL,
                featureClassName TEXT NOT NULL,
                definition TEXT NOT NULL,
                EGMFeatureClass TEXT NOT NULL,
                featureType TEXT NOT NULL,
                primitiveType TEXT NOT NULL,
                portrayalCriteria TEXT NOT NULL
                );''')
        cursor.execute('INSERT INTO KM1000_lake_area_07_2018_description VALUES (?,?,?,?
    ,?,?,?)', table)
        conn.commit()
        print ("FINISHED Inserting description for KM1000_lake_area")

        ##Manually add description of layer to gpkg for for KM1000_name_point
        FeatureClassName = "KM1000_name_point"
        Definition = "A geographic place on the earth, not normally appearing as a featu
    re on a map, but having a name that is required to be placed on a map."
        EGM_Feature_Class = "NameP"
        FeatureType = "Point"
        PrimitiveType = "Isolated node"
        PortrayalCriteria = "Cartographic text needed for named place at scale 1:1 000 0
    00 that cannot be put into attributes or features."
        attributeTable = "KM1000_name_point_07_2018_description_of_attributes"
        table = [1,FeatureClassName, Definition, EGM_Feature_Class, FeatureType, Primiti
    veType, PortrayalCriteria]
        cursor.execute('DROP TABLE IF EXISTS KM1000_name_point_07_2018_description')
        cursor.execute('''''CREATE TABLE KM1000_name_point_07_2018_description (
                id INTEGER CONSTRAINT m_pk PRIMARY KEY ASC NOT NULL,
                featureClassName TEXT NOT NULL,
                definition TEXT NOT NULL,
                EGMFeatureClass TEXT NOT NULL,
                featureType TEXT NOT NULL,
                primitiveType TEXT NOT NULL,
                portrayalCriteria TEXT NOT NULL
                );''')
        cursor.execute('INSERT INTO KM1000_name_point_07_2018_description VALUES (?,?,?,
    ?,?,?,?)', table)
        conn.commit()
        print ("FINISHED Inserting description for KM1000_name_point")

        ##Manually add description of layer to gpkg for KM1000_polbnd_area
        FeatureClassName = "KM1000_polbnd_area"
        Definition = "An area controlled by administrative authority."
        EGM_Feature_Class = "PolbndA"
        FeatureType = "Area"
        PrimitiveType = "Face"
        PortrayalCriteria = "Each administrative unit consists of one main area and occa
    sionally of one main area with exclave(s). Exclaves bigger than 3 km² included. If a co
    untry has national administrative levels below a country level, then the lowest level i
    n EU-
```

```
    countries is a level equivalent to NUTS3 level and in other countries the lowest level
    is comparable to this level."
428.        attributeTable = "KM1000_polbnd_area_07_2018_description_of_attributes"
429.        table = [1,FeatureClassName, Definition, EGM_Feature_Class, FeatureType, Primiti
    veType, PortrayalCriteria]
430.        cursor.execute('DROP TABLE IF EXISTS KM1000_polbnd_area_07_2018_description')
431.        cursor.execute('''''CREATE TABLE KM1000_polbnd_area_07_2018_description (
432.                id INTEGER CONSTRAINT m_pk PRIMARY KEY ASC NOT NULL,
433.                featureClassName TEXT NOT NULL,
434.                definition TEXT NOT NULL,
435.                EGMFeatureClass TEXT NOT NULL,
436.                featureType TEXT NOT NULL,
437.                primitiveType TEXT NOT NULL,
438.                portrayalCriteria TEXT NOT NULL
439.                );''')
440.        cursor.execute('INSERT INTO KM1000_polbnd_area_07_2018_description VALUES (?,?,?
    ,?,?,?,?)', table)
441.        conn.commit()
442.        print ("FINISHED Inserting description for KM1000_polbnd_area")
443.
444.        ##Manually add description of layer to gpkg for KM1000_polbnd_line(extra column
    added)
445.        FeatureClassName = "KM1000_polbnd_line"
446.        Definition = "A line of demarcation between controlled areas."
447.        EGM_Feature_Class = "POLBNDL"
448.        FeatureType = "Line"
449.        PrimitiveType = "Edge"
450.        PortrayalCriteria = "Boundary of an entity controlled by an administrative autho
    rity, this entity can be composed of several areas. All international boundaries. If a
    country has national administrative levels below a country level, then in EU countries
    all levels from country level to a level equivalent to NUTS3 are stored and in other co
    untries all levels from country level to a comparable level (i.e. LEVEL4 for CEEC count
    ries) are stored. This feature type is used also to close the administrative areas in t
    hose cases, when the location of the real international boundary is not stored on sea a
    rea."
451.        qualityCriteria ="International boundaries have to be geometrically consistent w
    ith topographical features (mainly the hydrographical ones). Geometrical consistency is
     recommended at lower level."
452.        attributeTable = "KM1000_polbnd_line_07_2018_description_of_attributes"
453.        table = [1,FeatureClassName, Definition, EGM_Feature_Class, FeatureType, Primiti
    veType, PortrayalCriteria, qualityCriteria]
454.        cursor.execute('DROP TABLE IF EXISTS KM1000_polbnd_line_07_2018_description')
455.        cursor.execute('''''CREATE TABLE KM1000_polbnd_line_07_2018_description (
456.                id INTEGER CONSTRAINT m_pk PRIMARY KEY ASC NOT NULL,
457.                featureClassName TEXT NOT NULL,
458.                definition TEXT NOT NULL,
459.                EGMFeatureClass TEXT NOT NULL,
460.                featureType TEXT NOT NULL,
461.                primitiveType TEXT NOT NULL,
462.                portrayalCriteria TEXT NOT NULL,
463.                qualityCriteria TEXT NOT NULL
464.                );''')
465.        cursor.execute('INSERT INTO KM1000_polbnd_line_07_2018_description VALUES (?,?,?
    ,?,?,?,?,?)', table)
466.        conn.commit()
467.        print ("FINISHED Inserting description for KM1000_polbnd_line")
468.
469.        ##Manually add description of layer to gpkg for KM1000_railrd_node
470.        FeatureClassName = "KM1000_railrd_node"
471.        Definition = "A stopping place for the transfer of passengers and/or freight."
472.        EGM_Feature_Class = "RailrdC"
```

```
473.        FeatureType = "Point"
474.        PrimitiveType = "Connected node"
475.        PortrayalCriteria = "Important main railway stations used for regular passenger
    traffic inside or near settlements."
476.        attributeTable = "KM1000_railrd_node_07_2018_description_of_attributes"
477.        table = [1,FeatureClassName, Definition, EGM_Feature_Class, FeatureType, Primiti
    veType, PortrayalCriteria]
478.        cursor.execute('DROP TABLE IF EXISTS KM1000_railrd_node_07_2018_description')
479.        cursor.execute('''''CREATE TABLE KM1000_railrd_node_07_2018_description (
480.                id INTEGER CONSTRAINT m_pk PRIMARY KEY ASC NOT NULL,
481.                featureClassName TEXT NOT NULL,
482.                definition TEXT NOT NULL,
483.                EGMFeatureClass TEXT NOT NULL,
484.                featureType TEXT NOT NULL,
485.                primitiveType TEXT NOT NULL,
486.                portrayalCriteria TEXT NOT NULL
487.                );''')
488.        cursor.execute('INSERT INTO KM1000_railrd_node_07_2018_description VALUES (?,?,?
    ,?,?,?,?)', table)
489.        conn.commit()
490.        print ("FINISHED Inserting description for KM1000_railrd_node")
491.
492.        ##Manually add description of layer to gpkg for KM1000_railrd_line
493.        FeatureClassName = "KM1000_railrd_line"
494.        Definition = "A rail or set of parallel rails on which a train or tram runs."
495.        EGM_Feature_Class = "RailrdL"
496.        FeatureType = "Line"
497.        PrimitiveType = "Edge"
498.        PortrayalCriteria = "Railway routes used for regular transportation of goods and
     passengers.Important industry railways can be included. Metro lines (= underground urb
    an railways), tramlines or streetcar lines inside city areas are excluded.Railways are
    represented by one line regardless of the number of tracks.Railway yards are excluded.
    Railway lines shorter than 2 km are excluded."
499.        attributeTable = "KM1000_railrd_line_07_2018_description_of_attributes"
500.        table = [1,FeatureClassName, Definition, EGM_Feature_Class, FeatureType, Primiti
    veType, PortrayalCriteria]
501.        cursor.execute('DROP TABLE IF EXISTS KM1000_railrd_line_07_2018_description')
502.        cursor.execute('''''CREATE TABLE KM1000_railrd_line_07_2018_description (
503.                id INTEGER CONSTRAINT m_pk PRIMARY KEY ASC NOT NULL,
504.                featureClassName TEXT NOT NULL,
505.                definition TEXT NOT NULL,
506.                EGMFeatureClass TEXT NOT NULL,
507.                featureType TEXT NOT NULL,
508.                primitiveType TEXT NOT NULL,
509.                portrayalCriteria TEXT NOT NULL
510.                );''')
511.        cursor.execute('INSERT INTO KM1000_railrd_line_07_2018_description VALUES (?,?,?
    ,?,?,?,?)', table)
512.        conn.commit()
513.        print ("FINISHED Inserting description for KM1000_railrd_line")
514.
515.        ##Manually add description of layer to gpkg for KM1000_reservoir_area
516.        FeatureClassName = "KM1000_reservoir_area"
517.        Definition = "A man-made enclosure or area formed for the storage of water"
518.        EGM_Feature_Class = "LakeresA"
519.        FeatureType = "Area"
520.        PrimitiveType = "Face"
521.        PortrayalCriteria = "Reservoirs larger than 0.5 km². Reservoirs being part of th
    e water network have to be topologically connected to watercourses."
522.        attributeTable = "KM1000_reservoir_area_07_2018_description_of_attributes"
```

```
523.        table = [1,FeatureClassName, Definition, EGM_Feature_Class, FeatureType, Primiti
    veType, PortrayalCriteria]
524.        cursor.execute('DROP TABLE IF EXISTS KM1000_reservoir_area_07_2018_description')

525.        cursor.execute('''''CREATE TABLE KM1000_reservoir_area_07_2018_description (
526.                id INTEGER CONSTRAINT m_pk PRIMARY KEY ASC NOT NULL,
527.                featureClassName TEXT NOT NULL,
528.                definition TEXT NOT NULL,
529.                EGMFeatureClass TEXT NOT NULL,
530.                featureType TEXT NOT NULL,
531.                primitiveType TEXT NOT NULL,
532.                portrayalCriteria TEXT NOT NULL
533.                );''')
534.        cursor.execute('INSERT INTO KM1000_reservoir_area_07_2018_description VALUES (?,
    ?,?,?,?,?,?)', table)
535.        conn.commit()
536.        print ("FINISHED Inserting description for KM1000_reservoir_area")
537.
538.        ##Manually add description of layer to gpkg for KM1000_road_line
539.        FeatureClassName = "KM1000_road_line"
540.        Definition = "An open way maintained for vehicular use"
541.        EGM_Feature_Class = "RoadL"
542.        FeatureType = "Line"
543.        PrimitiveType = "Edge"
544.        PortrayalCriteria = "Roads that form up a logical transportation network at a ma
    p scale 1:1 000000. Roads can be omitted for cartographic reasons in those areas where
    the road network is very dense. Low-
    class roads can be added if these roads are important routes in settlement structure. R
    oads are represented by one line regardless of the number of lanes or carriageways. Roa
    d lines shorter than 2 km are excluded. All European roads (E-roads) are included."
545.        attributeTable = "KM1000_road_line_07_2018_description_of_attributes"
546.        table = [1,FeatureClassName, Definition, EGM_Feature_Class, FeatureType, Primiti
    veType, PortrayalCriteria]
547.        cursor.execute('DROP TABLE IF EXISTS KM1000_road_line_07_2018_description')
548.        cursor.execute('''''CREATE TABLE KM1000_road_line_07_2018_description (
549.                id INTEGER CONSTRAINT m_pk PRIMARY KEY ASC NOT NULL,
550.                featureClassName TEXT NOT NULL,
551.                definition TEXT NOT NULL,
552.                EGMFeatureClass TEXT NOT NULL,
553.                featureType TEXT NOT NULL,
554.                primitiveType TEXT NOT NULL,
555.                portrayalCriteria TEXT NOT NULL
556.                );''')
557.        cursor.execute('INSERT INTO KM1000_road_line_07_2018_description VALUES (?,?,?,?
    ,?,?,?)', table)
558.        conn.commit()
559.        print ("FINISHED Inserting description for KM1000_road_line")
560.
561.        ##Manually add description of layer to gpkg for KM1000_builtup_area(typo in BEVP
    DF)
562.        FeatureClassName = "KM1000_builtup_area"
563.        Definition = "An area containing a concentration of buildings and other structur
    es."
564.        EGM_Feature_Class = "BuiltupA"
565.        FeatureType = "Area"
566.        PrimitiveType = "Face"
567.        PortrayalCriteria = '''''All built-
    up areas with equal or more than 50 000 inhabitants AND total size minimum 0.3 km². Min
    imum size of a discrete area: 0.3 km² (when the same built-
    up area is splitted to parts). Area 0.3 km² is used as only criteria when the number of
     inhabitants is unknown. Certain seamless (= compound) built-
```

```
    up areas can be split into separate parts with common borderlines if it is possible to
    attach a respective number of inhabitants (expressed by actual or class values) to each
     area separately. In that case all parts of this certain built-
    up area are represented as closed areas even if the number of inhabitants of a single p
    art is less than 50 000. Also actual names of each part can be stored. If it's not poss
    ible to separate the number of inhabitants, then this certain built-
    up area is stored unsplit as one area and names of the sub-
    areas can be stored separated with slash / like: Namex/Namey/Namez
568.        When a certain city is represented as several separated parts, then all these ar
    eas have the same name of this city and the same number of inhabitants is stored to eve
    ry part of this certain city. An area which does not fulfil the conditions named in the
     specs but is closed and surrounded by one or several other features of the coverage is
     called background area (= "hole"). Background areas or sparsely populated areas surrou
    nded by built-up areas smaller than 5 km² (inside built-
    up areas) are merged to the surrounding built-up areas.'''
569.        attributeTable = "KM1000_buildup_area_07_2018_description_of_attributes"
570.        table = [1,FeatureClassName, Definition, EGM_Feature_Class, FeatureType, Primiti
    veType, PortrayalCriteria]
571.        cursor.execute('DROP TABLE IF EXISTS KM1000_builtup_area_07_2018_description')
572.        cursor.execute('''''CREATE TABLE KM1000_builtup_area_07_2018_description (
573.                id INTEGER CONSTRAINT m_pk PRIMARY KEY ASC NOT NULL,
574.                featureClassName TEXT NOT NULL,
575.                definition TEXT NOT NULL,
576.                EGMFeatureClass TEXT NOT NULL,
577.                featureType TEXT NOT NULL,
578.                primitiveType TEXT NOT NULL,
579.                portrayalCriteria TEXT NOT NULL
580.                );''')
581.        cursor.execute('INSERT INTO KM1000_builtup_area_07_2018_description VALUES (?,?,
    ?,?,?,?,?)', table)
582.        conn.commit()
583.        print ("FINISHED Inserting description for KM1000_builtup_area")
584.
585.        ##Manually add description of layer to gpkg for KM1000_builtup_point
586.        FeatureClassName = "KM1000_builtup_point"
587.        Definition = "An area containing a concentration of buildings and other structur
    es."
588.        EGM_Feature_Class = "BuiltupP"
589.        FeatureType = "Point"
590.        PrimitiveType = "Isolated node"
591.        PortrayalCriteria = "All built-
    up areas with 1 000 – 50 000 inhabitants OR total size less than 0.3 km² (despite the n
    umber of inhabitants) Built-
    up areas which have less than 1000 inhabitants but are main villages or cities of the r
    egional/local administrative units are included. In that case it should be taken care t
    hat all regional/local administrative units have at least main village or city. If the
    number of inhabitants is not known, then the selection criterion is size less than 0.3
    km²."
592.        attributeTable = "KM1000_builtup_point_07_2018_description_of_attributes"
593.        table = [1,FeatureClassName, Definition, EGM_Feature_Class, FeatureType, Primiti
    veType, PortrayalCriteria]
594.        cursor.execute('DROP TABLE IF EXISTS KM1000_builtup_point_07_2018_description')

595.        cursor.execute('''''CREATE TABLE KM1000_builtup_point_07_2018_description (
596.                id INTEGER CONSTRAINT m_pk PRIMARY KEY ASC NOT NULL,
597.                featureClassName TEXT NOT NULL,
598.                definition TEXT NOT NULL,
599.                EGMFeatureClass TEXT NOT NULL,
600.                featureType TEXT NOT NULL,
601.                primitiveType TEXT NOT NULL,
602.                portrayalCriteria TEXT NOT NULL
```

```
                );''')
        cursor.execute('INSERT INTO KM1000_builtup_point_07_2018_description VALUES (?,?,?,?,?,?,?)', table)
        conn.commit()
        print ("FINISHED Inserting description for KM1000_builtup_point")

        ##Manually add description of layer to gpkg for KM1000_spring_node
        FeatureClassName = "KM1000_spring_node"
        Definition = "A natural outflow of water from below the ground surface."
        EGM_Feature_Class = "SpringC"
        FeatureType = "Point"
        PrimitiveType = "Connected node"
        PortrayalCriteria = "Springs that are considered as landmark by their location or size, or have a tourist interest and that are not related to the water network."
        attributeTable = "KM1000_spring_07_2018_description_of_attributes"
        table = [1,FeatureClassName, Definition, EGM_Feature_Class, FeatureType, PrimitiveType, PortrayalCriteria]
        cursor.execute('DROP TABLE IF EXISTS KM1000_spring_node_07_2018_description')
        cursor.execute('''''CREATE TABLE KM1000_spring_node_07_2018_description (
                id INTEGER CONSTRAINT m_pk PRIMARY KEY ASC NOT NULL,
                featureClassName TEXT NOT NULL,
                definition TEXT NOT NULL,
                EGMFeatureClass TEXT NOT NULL,
                featureType TEXT NOT NULL,
                primitiveType TEXT NOT NULL,
                portrayalCriteria TEXT NOT NULL
                );''')
        cursor.execute('INSERT INTO KM1000_spring_node_07_2018_description VALUES (?,?,?,?,?,?,?)', table)
        conn.commit()
        print ("FINISHED Inserting description for KM1000_spring_node")

        ##Manually add description of layer to gpkg for KM1000_watrcrs_area
        FeatureClassName = "KM1000_spring_node"
        Definition = "A natural or man-made flowing watercourse or stream."
        EGM_Feature_Class = "WatrcrsA"
        FeatureType = "Area"
        PrimitiveType = "Face"
        PortrayalCriteria = "Watercourse with width >= 500 m."
        attributeTable = "KM1000_watrcrs_area_07_2018_description_of_attributes"
        table = [1,FeatureClassName, Definition, EGM_Feature_Class, FeatureType, PrimitiveType, PortrayalCriteria]
        cursor.execute('DROP TABLE IF EXISTS KM1000_watrcrs_area_07_2018_description')
        cursor.execute('''''CREATE TABLE KM1000_watrcrs_area_07_2018_description (
                id INTEGER CONSTRAINT m_pk PRIMARY KEY ASC NOT NULL,
                featureClassName TEXT NOT NULL,
                definition TEXT NOT NULL,
                EGMFeatureClass TEXT NOT NULL,
                featureType TEXT NOT NULL,
                primitiveType TEXT NOT NULL,
                portrayalCriteria TEXT NOT NULL
                );''')
        cursor.execute('INSERT INTO KM1000_watrcrs_area_07_2018_description VALUES (?,?,?,?,?,?,?)', table)
        conn.commit()
        print ("FINISHED Inserting description for KM1000_watrcrs_area")

        ##Manually add description of layer to gpkg for KM1000_watrcrs_line
        FeatureClassName = "KM1000_watrcrs_line"
        Definition = "A natural or man-made flowing watercourse or stream."
        EGM_Feature_Class = "WatrcrsL"
```

```
        FeatureType = "Line"
        PrimitiveType = "Edge"
        PortrayalCriteria = "Watercourse with width >10-20 m and < 500 m."
        attributeTable = "KM1000_watrcrs_area_07_2018_description_of_attributes"
        table = [1,FeatureClassName, Definition, EGM_Feature_Class, FeatureType, Primiti
    veType, PortrayalCriteria]
        cursor.execute('DROP TABLE IF EXISTS KM1000_watrcrs_line_07_2018_description')
        cursor.execute('''''CREATE TABLE KM1000_watrcrs_line_07_2018_description (
                id INTEGER CONSTRAINT m_pk PRIMARY KEY ASC NOT NULL,
                featureClassName TEXT NOT NULL,
                definition TEXT NOT NULL,
                EGMFeatureClass TEXT NOT NULL,
                featureType TEXT NOT NULL,
                primitiveType TEXT NOT NULL,
                portrayalCriteria TEXT NOT NULL
                );''')
        cursor.execute('INSERT INTO KM1000_watrcrs_line_07_2018_description VALUES (?,?,
    ?,?,?,?,?)', table)
        conn.commit()
        print ("FINISHED Inserting description for KM1000_watrcrs_line")

        ##Insert description of attributes table of each layers to GeoPackage
        DescriptionOfAttributes = pd.read_csv("tablepage23.csv",nrows = 11)
        cursor.execute('DROP TABLE IF EXISTS KM1000_airfld_point_07_2018_description_of_
    attributes')
        DescriptionOfAttributes.to_sql("KM1000_airfld_point_07_2018_description_of_attri
    butes",conn, if_exists='append',index=False)
        conn.commit()
        print ("FINISHED Inserting description of attributes for KM1000_airfld_point")

        skiprows = list(range(12))
        DescriptionOfAttributes = pd.read_csv("tablepage23.csv",skiprows = skiprows)
        cursor.execute('DROP TABLE IF EXISTS KM1000_dam_line_07_2018_description_of_attr
    ibutes')
        DescriptionOfAttributes.to_sql("KM1000_dam_line_07_2018_description_of_attribute
    s",conn, if_exists='append',index=False)
        conn.commit()
        print ("FINISHED Inserting description of attributes for KM1000_dam_line")
        if os.path.exists("tablepage23.csv"):
            os.remove("tablepage23.csv")
        else:
            print("The file does not exist")

        DescriptionOfAttributes = pd.read_csv("tablepage24.csv",nrows = 8)
        cursor.execute('DROP TABLE IF EXISTS KM1000_elev_point_07_2018_description_of_at
    tributes')
        DescriptionOfAttributes.to_sql("KM1000_elev_point_07_2018_description_of_attribu
    tes",conn, if_exists='append',index=False)
        conn.commit()
        print ("FINISHED Inserting description of attributes for KM1000_elev_point")

        skiprows = list(range(9))
        DescriptionOfAttributes = pd.read_csv("tablepage24.csv",skiprows = skiprows)
        cursor.execute('DROP TABLE IF EXISTS KM1000_glacier_area_07_2018_description_of_
    attributes')
        DescriptionOfAttributes.to_sql("KM1000_glacier_area_07_2018_description_of_attri
    butes",conn, if_exists='append',index=False)
        conn.commit()
        print ("FINISHED Inserting description of attributes for KM1000_glacier_area")
        if os.path.exists("tablepage24.csv"):
            os.remove("tablepage24.csv")
```

```
709.        else:
710.            print("The file does not exist")
711.
712.        DescriptionOfAttributes = pd.read_csv("tablepage25.csv",nrows = 7)
713.        cursor.execute('DROP TABLE IF EXISTS KM1000_island_area_07_2018_description_of_a
    ttributes')
714.        DescriptionOfAttributes.to_sql("KM1000_island_area_07_2018_description_of_attrib
    utes",conn, if_exists='append',index=False)
715.        conn.commit()
716.        print ("FINISHED Inserting description of attributes for KM1000_island_area")
717.
718.        skiprows = list(range(8))
719.        DescriptionOfAttributes = pd.read_csv("tablepage25.csv",skiprows = skiprows)
720.        cursor.execute('DROP TABLE IF EXISTS KM1000_lake_area_07_2018_description_of_att
    ributes')
721.        DescriptionOfAttributes.to_sql("KM1000_lake_area_07_2018_description_of_attribut
    es",conn, if_exists='append',index=False)
722.        conn.commit()
723.        print ("FINISHED Inserting description of attributes for KM1000_lake_area")
724.        if os.path.exists("tablepage25.csv"):
725.            os.remove("tablepage25.csv")
726.        else:
727.            print("The file does not exist")
728.
729.        DescriptionOfAttributes = pd.read_csv("tablepage2627.csv",nrows = 8, usecols = [
    0,2,3])
730.        cursor.execute('DROP TABLE IF EXISTS KM1000_name_point_07_2018_description_of_at
    tributes')
731.        DescriptionOfAttributes.to_sql("KM1000_name_point_07_2018_description_of_attribu
    tes",conn, if_exists='append',index=False)
732.        conn.commit()
733.        print ("FINISHED Inserting description of attributes for KM1000_name_point")
734.
735.        skiprows = list(range(9)) + list(range(17,26))
736.        DescriptionOfAttributes = pd.read_csv("tablepage2627.csv",skiprows = skiprows, u
    secols = [0,2,3],encoding='latin1')
737.        cursor.execute('DROP TABLE IF EXISTS KM1000_polbnd_area_07_2018_description_of_a
    ttributes')
738.        DescriptionOfAttributes.to_sql("KM1000_polbnd_area_07_2018_description_of_attrib
    utes",conn, if_exists='append',index=False)
739.        conn.commit()
740.        print ("FINISHED Inserting description of attributes for KM1000_polbnd_area")
741.
742.        skiprows = list(range(17)) + list(range(21,26))
743.        DescriptionOfAttributes = pd.read_csv("tablepage2627.csv",skiprows = skiprows, u
    secols = [0,2,3],encoding='latin1')
744.        cursor.execute('DROP TABLE IF EXISTS KM1000_polbnd_line_07_2018_description_of_a
    ttributes')
745.        DescriptionOfAttributes.to_sql("KM1000_polbnd_line_07_2018_description_of_attrib
    utes",conn, if_exists='append',index=False)
746.        conn.commit()
747.        print ("FINISHED Inserting description of attributes for KM1000_polbnd_line")
748.
749.        skiprows = list(range(21))
750.        DescriptionOfAttributes = pd.read_csv("tablepage2627.csv",skiprows = skiprows, u
    secols = [0,2,3],encoding='latin1')
751.        cursor.execute('DROP TABLE IF EXISTS KM1000_railrd_node_07_2018_description_of_a
    ttributes')
752.        DescriptionOfAttributes.to_sql("KM1000_railrd_node_07_2018_description_of_attrib
    utes",conn, if_exists='append',index=False)
753.        conn.commit()
```

```
        header = ["Attribute","Definition","Value/Code or Example Value description"]
        DescriptionOfAttributes = pd.read_csv("tablepage28.csv",nrows = 3,names = header
    )
        DescriptionOfAttributes.to_sql("KM1000_railrd_node_07_2018_description_of_attrib
    utes",conn,if_exists='append',index=False)
        conn.commit()
        print ("FINISHED Inserting description of attributes for KM1000_railrd_node")
        if os.path.exists("tablepage2627.csv"):
            os.remove("tablepage2627.csv")
        else:
            print("The file does not exist")

        skiprows = list(range(3))
        DescriptionOfAttributes = pd.read_csv("tablepage28.csv",skiprows = skiprows, enc
    oding='latin1')
        cursor.execute('DROP TABLE IF EXISTS KM1000_railrd_line_07_2018_description_of_a
    ttributes')
        DescriptionOfAttributes.to_sql("KM1000_railrd_line_07_2018_description_of_attrib
    utes",conn, if_exists='append',index=False)
        conn.commit()
        print ("FINISHED Inserting description of attributes for KM1000_railrd_line")
        if os.path.exists("tablepage28.csv"):
            os.remove("tablepage28.csv")
        else:
            print("The file does not exist")

        DescriptionOfAttributes = pd.read_csv("tablepage2930.csv",nrows = 10)
        cursor.execute('DROP TABLE IF EXISTS KM1000_reservoir_area_07_2018_description_o
    f_attributes')
        DescriptionOfAttributes.to_sql("KM1000_reservoir_area_07_2018_description_of_att
    ributes",conn, if_exists='append',index=False)
        conn.commit()
        print ("FINISHED Inserting description of attributes for KM1000_reservoir_area")


        skiprows = list(range(11)) + list(range(21,32))
        DescriptionOfAttributes = pd.read_csv("tablepage2930.csv",skiprows = skiprows, e
    ncoding='latin1')
        cursor.execute('DROP TABLE IF EXISTS KM1000_road_line_07_2018_description_of_att
    ributes')
        DescriptionOfAttributes.to_sql("KM1000_road_line_07_2018_description_of_attribut
    es",conn, if_exists='append',index=False)
        conn.commit()
        print ("FINISHED Inserting description of attributes for KM1000_road_line")

        ##(typo in BEVPDF)
        skiprows = list(range(21))
        DescriptionOfAttributes = pd.read_csv("tablepage2930.csv",skiprows = skiprows, e
    ncoding='latin1')
        cursor.execute('DROP TABLE IF EXISTS KM1000_builtup_area_07_2018_description_of_
    attributes')
        DescriptionOfAttributes.to_sql("KM1000_builtup_area_07_2018_description_of_attri
    butes",conn, if_exists='append',index=False)
        conn.commit()
        print ("FINISHED Inserting description of attributes for KM1000_builtup_area")
        if os.path.exists("tablepage2930.csv"):
            os.remove("tablepage2930.csv")
        else:
            print("The file does not exist")

        DescriptionOfAttributes = pd.read_csv("tablepage31.csv",nrows = 10)
```

```
        cursor.execute('DROP TABLE IF EXISTS KM1000_builtup_point_07_2018_description_of_attributes')
        DescriptionOfAttributes.to_sql("KM1000_builtup_point_07_2018_description_of_attributes",conn, if_exists='append',index=False)
        conn.commit()
        print ("FINISHED Inserting description of attributes for KM1000_builtup_point")

        skiprows = list(range(11))
        DescriptionOfAttributes = pd.read_csv("tablepage31.csv",skiprows = skiprows, encoding='latin1')
        cursor.execute('DROP TABLE IF EXISTS KM1000_spring_node_07_2018_description_of_attributes')
        DescriptionOfAttributes.to_sql("KM1000_spring_node_07_2018_description_of_attributes",conn, if_exists='append',index=False)
        conn.commit()
        print ("FINISHED Inserting description of attributes for KM1000_spring_node")
        if os.path.exists("tablepage31.csv"):
            os.remove("tablepage31.csv")
        else:
            print("The file does not exist")

        DescriptionOfAttributes = pd.read_csv("tablepage32.csv")
        cursor.execute('DROP TABLE IF EXISTS KM1000_watrcrs_area_07_2018_description_of_attributes')
        DescriptionOfAttributes.to_sql("KM1000_watrcrs_area_07_2018_description_of_attributes",conn, if_exists='append',index=False)
        conn.commit()
        print ("FINISHED Inserting description of attributes for KM1000_watrcrs_area")
        if os.path.exists("tablepage32.csv"):
            os.remove("tablepage32.csv")
        else:
            print("The file does not exist")

        DescriptionOfAttributes = pd.read_csv("tablepage33.csv")
        cursor.execute('DROP TABLE IF EXISTS KM1000_watrcrs_line_07_2018_description_of_attributes')
        DescriptionOfAttributes.to_sql("KM1000_watrcrs_line_07_2018_description_of_attributes",conn, if_exists='append',index=False)
        conn.commit()
        print ("FINISHED Inserting description of attributes for KM1000_watrcrs_line")
        if os.path.exists("tablepage33.csv"):
            os.remove("tablepage33.csv")
        else:
            print("The file does not exist")

        ##List all tables in the database
        cursor.execute("SELECT name FROM sqlite_master WHERE type='table';")
        tables = cursor.fetchall()
        for table in tables:
            print (table)

        sg.Popup('Program Finished','Use DB Browser for SQLite to check the results.')
```