Hints for effective use of the NLS Topo GeoPackage

Users are adviced to use the manuals of the client programs as a primary source when learning to use GeoPackage in the most effective way. Those manuals are most up-to-date and they are written by authors who know the specific details of their implementations. That GeoPackage is an OGC standard guarantees that any client program can perform the basic operations with GeoPackage but implementations differ and thereafter also the best workflows.

The best resource for finding information about the GeoPackage standard is the dedicated web site http://www.geopackage.org . That site provides also links to the corresponding page on the OGC:n standard site and to the GitHub-project that is the place where the development of the future versions and extensions of the standard happens. The issue tracker for the published parts of the standard is also in GitHub. Because GeoPackage in built on top of SQLite the documentation in https://sqlite.org/index.html is also relevant for GeoPackage developers.

The purpose of the following advice is to help users to adopt GeoPackage as a delivery media of the topographic database of the National Land Survey of Finland. The document is based on experiences from the initial tests of the NLS Topo GeoPackage in April 2020 and it is provided as is.

General advice

1- Don’t add too many features into the map

Some tables of the NLS Topo GeoPackage are rather large. Table ”maastokuvionreuna” (borders of the terrain polygons) contains more than 10 million linestrings, contour line layer ”korkeuskayra” has 6.8 million linestrings, and the building table ”rakennus” has 5.4 million polygons. If any GIS software tries to open and render all the features from so large tables on the map it will be slow. User should set a scale limit for the large layers so that only reasonable number of features will be rendered on the screen. Small tables do not need scale limits, but the rendering of contour lines should be stopped about at scale 1:50000. Scale limit approach is not good for the lake table ”jarvi”. It is probably better to post-process the data by adding a new attribute to the table and fill it with the areas of the lake polygons. This way it is possible to render only the biggest lakes at small map scales.

2. Your software may not know how to use the spatial index

All the tables in the NLS Topo GeoPackage have spatial indexes. Spatial index enables to select features fasts within a given bounding box. However, the existence of the spatial index does not guarantee that queries are getting any faster than without the index. The reason for this is that the R*Tree index system is published as an extension to the GeoPackage standard and client programs do not need to support it. If you experience poor performance when using the largest NLS Topo tables directly from the GeoPackage database it is better to convert data into some other format that suits better for your software. The workflow is then the same than when using the NLS Topo GML files. Advantage for a

user is that it is enough to download one GeoPackage file instead of 3000 GML files.

Software specific notes

ESRI products

ESRI ArcMap and ja ArcGIS Pro support GeoPackage best as a data transfer format. Both programs can add layers directly from GeoPackage into workspace but while this works fine for small tables a much better performance is achieved if the largest tables are converted first into some native ESRI format, for example File Geodatabase. GeoPackage layers cannot be edited directly with ESRI products. However, it is possible to edit layers in other formats and save the result into a new GeoPackage database.

Source: ArcGIS-blog https://www.esri.com/arcgis-blog/products/product/data-management/how-to-use-ogc-geopackages-in-arcgis-pro/

MapInfo

MapInfo Pro and MapInfo Pro Advanced support reading, editing, and creating GeoPackage feature tables.

Source https://www.geopackage.org/implementations.html

QGIS

QGIS is using GeoPackage as the default vector format since program version 3.0. QGIS deals well even with the largest tables of the NLS Topo GeoPackage, provided that user sets a scale limit that prevents rendering too many features at small scales.

Source: https://qgis.org/fi/docs/index.html

FME

FME reads and writes GeoPackage. The required license version is ”Desktop Professional” or higher. FME version 2020.0 contains a few useful improvements in the way how new GeoPackage databases are created but also at least version 2018.2 creates GeoPackages which are 100% standard compliant.

Source: https://docs.safe.com/fme/html/FME_Desktop_Documentation/FME_ReadersWriters/ogcgeopackage/ogcgeopackage.htm

GDAL

GDAL-utility program ogr2ogr can be used for reading, editing, and writing vector data in GeoPackage format. GDAL creates 100% standard compliant results. GeoPackages which are created with GDAL contain an additional metadata table that helps QGIS program to make the initial loading of layers faster. This additional table does not have any effect on other GeoPackage readers. Everything that can be done with the ogr2ogr executable is also possible to implement programmatically by utilizing the GDAL library and the language bindings that are available for C++, C#, Python, and Java.

Source: https://gdal.org/drivers/vector/gpkg.html

GeoServer

GeoServer can use GeoPackage as a performant data source for WMS and WFS services.

Source: https://docs.geoserver.org/latest/en/user/data/vector/geopkg.html

MapServer

MapServer can use GeoPackage as a performant data source for WMS and WFS services. The connection to GeoPackage is made through GDAL/OGR.

Source: https://www.mapserver.org/input/vector/ogr.html

Notes about the speed of GeoPackage database

GeoPackage is fundamentally just a normal SQLite database with partly fixed structure. SQLite is a serverless database that is widely used and fast, but like with all databases, queries from large tables without appropriate indexes are slow. A few examples of SQL queries from the NLS Tope GeoPackage follows.

Example 1. Selecting one feature by an unindexed attribute vs. by an attribute with an index. The former query is 250 times slower.

SELECT * FROM rakennus WHERE fid=5000001;

Execution time: 0.032 sec

SELECT * FROM rakennus WHERE mtk_id=1150058247;

Execution time: 8.505 sec

Example 2. Selecting features within a 1 km x 1 km sized bounding box. Query is finding 284 features out of 5462918. First version does not utilize spatial index (R*Tree-index), the second version does use the R*Tree index in a sub-query. First query version is 60 times slower.

Notice: The ST_Min/Max functions are not available in all environments and implementations and their speeds differ.

SELECT * FROM rakennus

WHERE ST_MaxX(geom)>=633070 AND ST_MinX(geom)<=634070

AND ST_MaxY(geom)>=6826718 AND ST_MinY(geom)<=6827718;

Execution time: 8.468 sec

SELECT * FROM rakennus

WHERE fid IN

(SELECT id FROM rtree_rakennus_geom

 WHERE maxX>=633070 AND minX<=634070

 AND maxY>=6826718 AND minY<=6827718);

Execution time: 0.135 sec

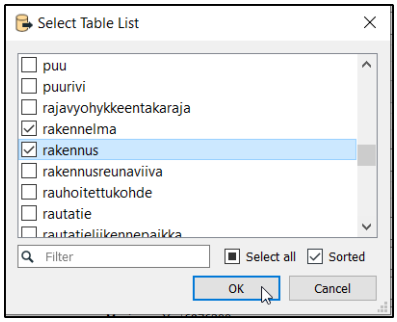
Timings are measuring just the time that is needed for getting data out of GeoPackage database. Programs are also doing a variable amount of processing for rendering the data on a map and the speed that is experienced by a user is slower.

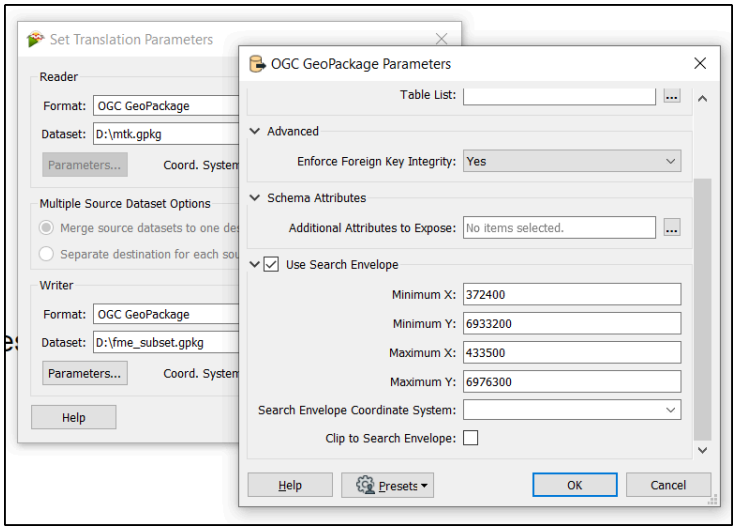
Making GeoPackage database smaller

If user do not need topographic data from the whole Finland of if some tables which are included in the NLS Topo GeoPackage are unnecessary it is possible to reduce the size of the database file by dropping tables or by clipping data by a bounding box. This saves disk space but it is also possible that programs which do not know how to utilize GeoPackage in an optimal way will be faster.

Selecting tables and clipping data by bounding box with FME Quick Translator

FME Quick Translator program has a graphical user interfase for selecting which tables will be copied into a new GeoPackage. It is also possible to define a clipping bounding box with minimum and maximum X and Y coordinates. This example copies only buildings and constructions which are located within the bounding box of the municipality of Saarijärvi.





Selecting tables and clipping data by bounding box with GDAL ogr2ogr program

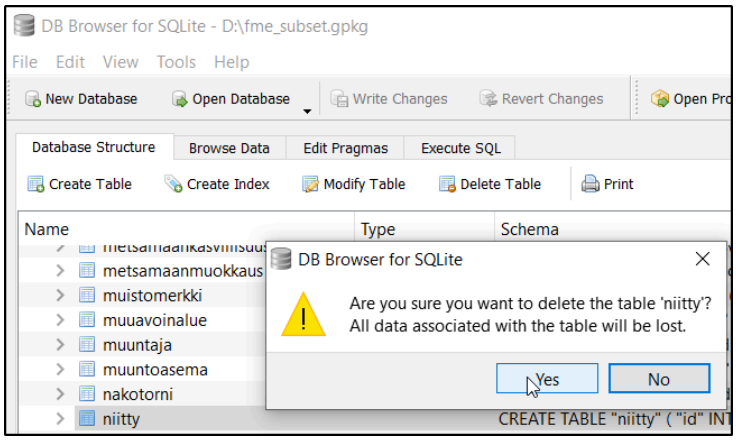
This command creates a new GeoPackage database that contains the buildings and constructions of Saarijärvi with an open source GDAL utility program ogr2ogr.

ogr2ogr -f gpkg -spat 372400 6933200 433500 6976300

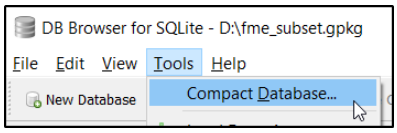
gdal_rakennus_leikattu.gpkg mtk.gpkg rakennus rakennelma

Dropping tables and compacting a GeoPackage database

Deleting unnecessary tables from a GeoPackage database is freeing space. Tables are deleted or dropped with a standard SQL statement ”DROP TABLE”. For dropping tables any SQLite client or method that can used if it just can connect the database and execute SQL commands. With an open source SQLite client ”DB Browser for SQLite” user do not even need to know the SQL syntax but tables can be dropped by using an entry from a menu that opens with a right-click.



Dropping tables from SQLite database makes the space free for other needs but it does not make the physical file on disk any smaller. For compacting the database and freeing disk space another SQL command ”VACUUM” must be fired. Also this task can be done with DB Browser without knowing SQL.



User who prefers command line over GUI can drop tables and vacuum the database with another GDAL utility ”ogrinfo”. It is worth noticing the VACUUM is actually writing a new complete database file and with multi-gigabyte databases the operation takes some time.

ogrinfo -sql "DROP TABLE niitty" mtkmaasto.gpkg
INFO: Open of `mtkmaasto.gpkg'
using driver `GPKG' successful.

ogrinfo -sql "VACUUM" mtkmaasto.gpkg
INFO: Open of `mtkmaasto.gpkg'
using driver `GPKG' successful.

Making attribute queries faster by creating a new index

Let’s return to example 1 and the slow query that selects one feature by attribute “mtk_id”. If there is a need to do these queries frequently user can create a new attribute index with their favorite SQL client.

CREATE INDEX "rakennus_mtk_id_idx"
ON "rakennus"

("mtk_id");

Repeat the test and compare the speed

SELECT * FROM rakennus WHERE mtk_id=1150058247;
Execution time: 0.043 sec (was 8.505 sec)