GeoPackage: Unifying Modeling and Simulation with Mission Command Geospatial Data

Kevin Bentley
Cognitics, Inc.
Boise, Idaho
kbentley@cognitics.net

Michala Hill
Leidos
Orlando, Florida
MICHAL.L.HILL@leidos.com

Ronald Moore
Leidos
Orlando, Florida
RONALD.G.MOORE@leidos.com

Mark Johnson
Optimal Solutions and Technologies (OST) Inc.
Orlando, Florida
mark.d.johnson302.ctr@mail.mil

ABSTRACT

OpenFlight™ (a trademark of Presagis, Inc.), the de-facto standard for 3D models and terrain for simulation and training, has been in common use for decades. Despite revisions and expansion over the years, the format remains largely unchanged.

The OpenFlight format does not lend itself to storage of abstract feature data and relationships between 3D visual objects and related abstract objects. A typical OpenFlight terrain database or 3D model built for one Image Generator (IG) may lack collision volumes or trafficability rules needed by another system. The OpenFlight database cannot preserve the relationship of polygons to the source features they were built from.

Modern SAFs and specific gaming systems all require conceptually different data models, meaning many output files and formats are required to support a confederate of training and simulation platforms.

Meanwhile, the Open Geospatial Consortium has adopted the GeoPackage format for Geographic Information Systems (GIS) data. GeoPackage provides a data model and consistency across applications. GeoPackage is built on SQLite, an open source, self-contained, cross-platform embedded database engine. As such, GeoPackage is ideal for expansion in support of the simulation industry. By storing abstract features with relationships to their specific counterparts, a single GeoPackage file could be used to exchange correlated data for simulation systems as well as mission command and GIS applications. A single feature stored in a GeoPackage file can have a relationship to a 3D model used in a legacy image generator and also a model designed for use in a gaming system or SAF.

This paper reports on SE Core's investigation into expanding on the GeoPackage standard for use in simulation and training. We present a demonstration where a single database with visual and abstract objects can provide all the intermediate data necessary to create database products for use in production.

ABOUT THE AUTHORS

Kevin Bentley is the founder and president of Cognitics, Inc. He has over 20 years of experience developing software for GIS, simulation, and games. He currently is the principal investigator for a Phase II SBIR focused on research and development of technologies that improve the representation of transportation models that integrate with terrain in simulation systems. He has over 20 years of experience developing commercial and open-source software applications and libraries for GIS, simulation, and video games. Mr. Bentley was the principal investigator for a Phase II, two Phase II enhancements, and a Phase III SBIR focused on research and development of technologies that improve the representation of transportation and hydrography models that integrate with terrain in simulation systems.

Michala Hill is a Senior GIS Analyst on SE Core. She has conducted geospatial research and analysis for over 15 years in such fields as ecology, conservation, agriculture, and modeling and simulation. Specialties include process development and improvement, and technical research. She holds a Master of Science degree in Geography from the University of Florida and is a certified GIS Professional (GISP).

Mark Johnson – Mr. Johnson is a senior visual systems engineer at OST in Orlando, FL. He is currently supporting PEO STRI as a SETA contractor, a technical expert in visual systems, synthetic environments, software development and architecture. Mr. Johnson received his undergraduate degree in computer science from Utah State University. He has been working with simulation and training systems since 1987 and has more than 35 years of experience in software and system design.

Ronald G. Moore is currently the Chief Architect on SE Core. He has over 30 years of experience in the simulation and training industry with expertise in software development, computer graphics, computer image generation, simulation geospatial database production, video, audio, sound simulation, and PC and console game development. His previous assignments include Senior Systems Engineer at SAIC; Research Scientist at E&S; Chief Technology Officer at Infogrames, and Lead Software Engineer at Boeing. Ron holds a BSE from BYU.

GeoPackage: Unifying Modeling and Simulation with Mission Command Geospatial Data

Kevin Bentley
Cognitics, Inc.
Boise, Idaho
kbentley@cognitics.net

Michala Hill
Leidos
Orlando, Florida
MICHAL.L.HILL@leidos.com

Ronald Moore
Leidos
Orlando, Florida
RONALD.G.MOORE@leidos.com

Mark Johnson
Optimal Solutions and Technologies (OST) Inc.
Orlando, Florida
mark.d.johnson302.ctr@mail.mil

INTRODUCTION

As simulation advances technologically, the synthetic world becomes nearer to the natural world. This means that the data used for simulation will become practically indistinguishable from the data used for Mission Command, Intelligence, and Mapping. However, while there are many Government Off the Shelf (GOTS), Commercial Off the Shelf (COTS), and Open Source tools that process, refine, analyze, and visualize mission command and Geographic Information Systems (GIS) data, both the tools and the data remain isolated from simulation because the relationships between the features are lost in the process of creating the simulation data from the GIS data. This problem stems from the fact that the formats used in simulation usually have limited mechanisms to store the GIS data, and very rarely have a way to represent the relationships between the simulation and GIS data. This makes it difficult to interchange data between simulation systems. In practice, each simulation system has variations in the data model it requires, and even when two different simulation systems can share the same format (e.g. OpenFlight), they often require vendor specific extensions or even different content within the format. Thus, data is frequently specialized for each system even when the same format is used. The formats themselves lack the ability to store abstract data that can be specialized by each system rather than during the database generation process.

The lack of traceability between source data, abstract representations of the data, and the concrete representation intended for each simulation means when a small change is needed in the source GIS data, entire datasets must be rebuilt because the simulation data has an unknown lineage. A polygon may have been produced by a combination of GIS data, geospecific imagery, geotypical textures, sensor data, and more. If the relationships between those features are preserved, then inspection, analysis, and modification can be performed without rebuilding the entire dataset.

For example, some systems may require data for physics based rendering or damage simulation, while others require collision data at different resolutions (for example, one may require simple cube models while others support only convex 3D polygons). Many systems are capable of rendering terrain directly from GIS data, but still need 3D models for buildings. Some systems extrude simple buildings from footprints, others support buildings with interiors.

This paper describes a solution to these problems by proposing a set of enhancements and standards definition to:

- Integrate the needs of the simulation community with the existing GeoPackage format to align simulation data and formats with the GIS and Mission Command community.
- Increase the ability to reuse and incrementally process simulation data.
- Optimize the performance of simulation databases while ensuring the integrity by embedding relational database concepts in the file format itself.

BACKGROUND

To address the above challenges, we investigated the creation of extensions and enhancements to the GeoPackage standard, leveraging existing GeoPackage extensions when possible.

GeoPackage is an open, standards based file that supports geospatial raster, vector data, and user extensions. The Open Geospatial Consortium adopted the version 1.0 of the standard in 2014, and it has been updated since then.

GeoPackage is more than just a file format, it is a self-contained database built on the SQLite engine. As such, each file is self-describing, with built-in optimizations and integrity behavior. Each data table can have customized indexes for application agnostic performance tuning.

Using the trigger functionality in a GeoPackage file, business rules, data validation, and references between different records within the same file can be enforced. A trigger allows changes in one record to trigger user specified operations. Commonly one feature is derived from a different feature. For example, a polygon can represent a treed tract. During the database production, point features may be created to place tree models in the terrain. Using triggers, the derived point features can automatically be changed or removed when the original feature is modified, as shown in Figure 1.

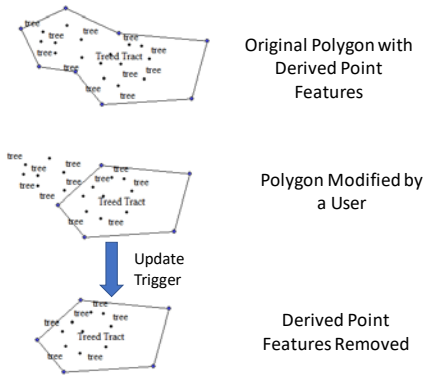

Figure 1 - Update Trigger Example. A change made by a user in a GIS application can automatically delete or modify features that derived from the original feature. This gives a user the ability to modify a terrain database through existing GIS applications, and have that change automatically and incrementally propogate through the dabase.

Similarly, a delete trigger can automatically delete derived features, or prevent derived features from being deleted directly, to preserve the integrity of the database. For example, if feature B is derived from feature A, a trigger can be used to prevent feature B from being directly modified or deleted, but automatically update or delete feature B if feature A is updated or deleted.

Triggers can be complex, leveraging functions supplied by SQLite, Spatialite, and GeoPackage. If a simulation specific extension to GeoPackage is developed, triggers can be written to modify the geometry or attribution of the simulation terrain when a change to a raster or feature level is made by a GIS application.

When a GeoPackage file is created or updated, a specialized index or integrity trigger can be created specifically for the data it contains.

SQLite is an Open Source, cross platform, embeddable relational database engine. SQLite is commonly used as an internal data store for mobile and desktop applications including the Google Chrome web browser. It has been carefully tuned for reliability and performance. For every line of code in SQLite, there are 745 lines of code in the test suite used to ensure the integrity of the engine.

With GeoPackage being built on the SQLite database engine there are performance benefits, especially when it comes to storage and retrieval of small data. Benchmarks of SQLite have shown that when storing small (around 10 kilobytes) files, SQLite performs up to 35% faster and uses up to 20% less space than using the filesystem (35% Faster Than The Filesystem, 2017). In simulation applications, small data such as vectors, vertex buffers, and low level of detail raster images (e.g. thumbnails) should benefit from GeoPackage compared to storing the data as individual files.

For these reasons, it is unlikely that any new file format developed in the simulation community can approach the performance, flexibility, and reliability of SQLite.

Adding functionality without breaking backwards compatibility is simple because GeoPackage is built on SQLite. New database tables (i.e. those not described in the existing standard) that reference spatial data can easily be added.

GeoPackage also can add new types of geometry through a well-defined extension specification, allowing the extension of new functionality through binary SQLite modules loaded at runtime.

GeoPackage is currently supported by several GIS applications and libraries including: ArcGIS, Global Mapper, GDAL, Safe FME, GeoServer, and QGIS (via GDAL).

The Army Geospatial Center has recognized the need for a single format to store Geospatial data across multiple GEOINT applications:

"If you have three or four applications that use data in different formats, you have to store that geospatial data several different ways on the device. But you can't do that, so they have come up with a standard called GeoPackage, which allows you to store all the data once, and have it used by multiple applications," he said. "This is a game-changer that we worked with the Open Geospatial Consortium (OGC) on, moving it through the process of approval as an OGC standard very quickly. It's a game changer not only for the Army, but also for industry," [Dr. Joseph Fontanella, head of the Geospatial Research Laboratory] added[1].

This same need has long applied to the simulation community. Creating a GeoPackage extension specialized for the simulation community has the potential to revolutionize the process of creating, maintaining, distributing, and deploying datasets across a variety of simulation platforms.

Typical data used in simulation includes: GIS vector, imagery, and elevation data, 3D terrain mesh, geospecific and geotypical models, geotypical terrain textures, moving models, behavioral data, physics data, and more. Our investigation aimed to determine the feasibility of using additions and extensions to GeoPackage to store these types of data.

INVESTIGATION

The intended use of the extensions to GeoPackage is to combine mission command GIS data with simulation data in a way that preserves logical and topological relationships, and provides for high performance queries and editing, while preserving portability between applications. Our investigation focused on these use cases.

To determine the level of effort and viability of a GeoPackage extension for use in simulation, we first created a schema to apply to an existing GeoPackage file containing vector and raster data. This schema was then populated using simulation data, including OpenFlight models and textures. Additionally, several scripts were created to populate an example database, establish relationships, and populate them into the 'enhanced' GeoPackage file.

This effort does not intend to define the specific data model and schema for use in simulation, but to provide an example and a viability test. Just as the GeoPackage specification does not define a data model or data dictionary for GIS features, we did not presume to define a specific proposed set of extensions at this time. An official proposal to extend the GeoPackage format must be an effort that includes the larger GeoPackage community.

Our end to end viability test includes a reference GIS dataset in GeoPackage format, a Structured Query Language (SQL) schema design, scripts to populate the simulation specific data as well as sample data including relationships, services, behaviors, and alternate representations.

[1] Quote from http://www.kmimediagroup.com/gif/424-articles-gif/from-topographic-to-geospatial/6305-from-topographic-to-geospatial

Relationships

Relationships are an important part of our design. Relationships are what allow us to tie GIS and Mission Command data to the simulation specific data that is often derived from GIS features. In the above treed tract example, a relationship would be established between the original areal feature and the scattered point features derived from it.

We created a schema that defines an enumeration of relationship types, which can be expanded as desired. For the above example, the 'SCATTERED_FROM' relationship exists from the point feature to the areal feature.

Relationships may also exist between features other than GIS. In the above case, there would also be a relationship between an external reference to a geotypical model in polygon terrain.

Alternate Representations

Features used in mission command, GIS, and Simulation frequently have multiple representations. For example, a school might be represented as a GIS point feature, a GIS footprint feature, a stylized map icon, or multiple 3D models (e.g. with interior walls or without). Using GeoPackage we can associate features with all forms of representation. In GIS and simulation applications, these forms of representation are usually chosen through rules that are unique to each system. By storing the alternate forms of representation inside the GeoPackage database, the data becomes self-describing. Self-describing data insures consistency across programs, platforms, and applications, and simplifies the terrain database generation process.

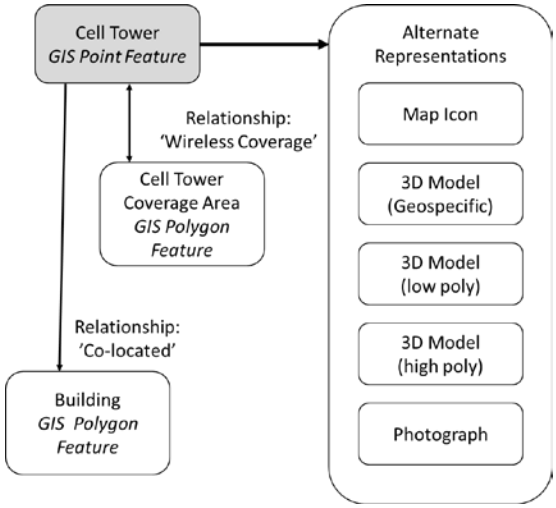

Figure 2 Alternate Representations and Relationships between Features

Services

Modeling megacities, patterns of life, and socio-cultural factors requires information about services that are provided by features with geospatial characteristics. For example, a power substation provides grid electricity. A cell tower provides network coverage. The services provided by a feature may be multi-dimensional, and not easily defined in GIS feature attributes. The combination of GIS features (stored as standard GeoPackage features) defined services, and relationships allows features providing any number of services to be efficiently located using indexed SQL queries.

Behaviors and Entities

While behaviors and entities may not have fixed geospatial coordinates, they frequently depend or interact with them. Entities may have multiple behaviors, and those behaviors may be dependent on features and/or services. Our goals in this experiment are to create a complete simulation database, so we included this information as part of the schema. In our experiment, we modeled an autonomous electric vehicle as an entity, which has driving behaviors (e.g. stopping for pedestrians) as well as recharging from an electric vehicle (EV) charging station. Figure 3 illustrates a simplified diagram of these relationships.

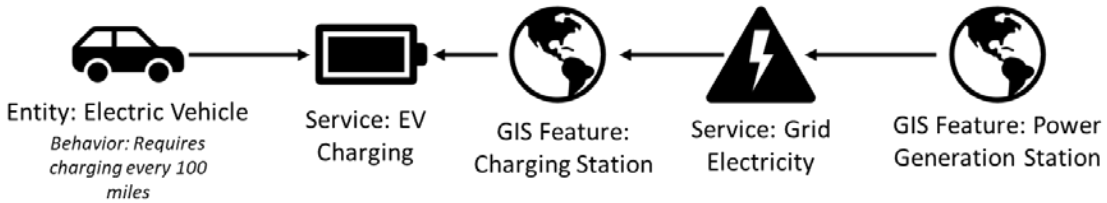

Figure 3 - Example Behaviors, Entities, Services, and GIS Feature Relationship

Comparisons to and Relationship with Existing formats

Common Database Format (CDB)

The CDB is an Open Geospatial Consortium (OGC) standard for storing and transmitting simulation data, optimized for runtime use in simulation applications. It provides a structure and data model including multiple levels of detail. The CDB schema defines a file system structure for storing large simulation datasets. Rasters are tiled by level of detail, and are stored as individual files. Vector data is stored as ShapeFiles and many models are stored as OpenFlight.

The core ideas of the CDB include defining a structured repository for GIS data, storage of pre-built levels of detail, and a built-in schema to support versioning and dynamic updates (Freeman & Srouji, 2017).

Our proposed simulation extensions to GeoPackage is not proposed as a replacement or alternative to the CDB. In fact, the simulation enhanced GeoPackage file could be used inside a CDB dataset, potentially replacing Shapefiles, OpenFlight files, and even GeoTIFF files. Or, as suggested by Freeman & Srouji (2017), a GeoPackage container could be used in place of a filesystem to store CDB formatted simulation data.

Layered Terrain Format (LTF)

The Layered Terrain Format (LTF) was designed specifically to support synthetic environments while optimizing disk space and bandwidth usage when streaming (Layered Terrain Format ; Format Specification, Schema, and Layer Profiles, 2016). It was designed as an improvement over legacy formats such as ShapeFiles that can be used on multiple simulation systems.

LTF supports multiple layers of data including vector features, raster data, point clouds, polygonal geometry, and many other data types used in simulation. In this regard, the LTF format covers many of the same concepts and data types as those we propose to extend for GeoPackage.

An important difference between LTF and GeoPackage is the use cases they are designed for. LTF is optimized as a transmittal format to be extremely compact, optimizing the use of bandwidth and storage space. LTF does not seem to be designed for edit-in-place, or integration with existing GIS applications. With the proposed extensions to GeoPackage, LTF files could be generated as an output from a GeoPackage file, using the GeoPackage capabilities to combine several datasets into a high efficiency LTF file.

Proof of Concept Design

The design for our proof of concept is meant to determine the viability of using an enhanced GeoPackage file for use in simulation. Our scope is not intended to determine every aspect of the concept, or to create a proposed specification or standard. Rather, our goal is to identify capabilities, limitations, and general performance of the concept.

To that end, we started with a reference GeoPackage file, created by exporting an existing SE Core ESRI File Geodatabase to a GeoPackage file using ESRI's ArcCatalog application. From there we applied an SQL script that contained the schema we designed. A simplified diagram of this schema is visible in Figure 4. Next, we developed an importer application to read an existing SE Core OpenFlight terrain tile and textures. The importer application populated our virtual simulation tables with polygons and textures.

To demonstrate the visualization of the virtual simulation data, we developed a plugin for OpenSceneGraph that can read the simulation enhanced GeoPackage file, and render the textured terrain file using the osgviewer application. Simultaneously, we used the ESRI ArcMap application to display the GIS features in the GeoPackage file. As an added verification step, we inserted a GeoPackage geometry column in the mesh table. The geometry column allowed ArcMap to render portions of the virtual table in the GIS table. While this data was redundant, for the experiment it allowed the mixture of GIS and simulation to be viewed in one application.

Test and demonstration of the remaining tables required some manual database entries. An additional SQL script applied these entries based on manual analysis of the GIS and virtual data. Example services, behaviors, and entities were also created for the SQL script.

SCHEMA

Because we had an abundance of correlated GIS and matching virtual terrain data, the data model for the virtual use case was the most detailed part of our investigation. We created a schema that can store and retrieve the visual polygon representation of an entire terrain database. We broke down the tables in the schema using a scene-graph model. This model was chosen for the experiment for several reasons. One was because it allowed us to create an end-to-end proof of concept (i.e. populating a GeoPackage file and rendering directly from it). Another reason is that the SQLite engine supports rectangular trees (R-Trees), which are spatial indexes that work in multiple dimensions. R-Trees are usually used in GeoPackage and Spatialite for 2D spatial indexing, but the R-Tree itself can support arbitrary dimensionality. If 3D R-Trees are used, it has the potential to be used for applications such as collision detection and data paging for very large databases.

Definition of the data models for some tables (including entities and behaviors) was beyond the scope of our effort, so we defined them as having a large text column in the database, using simple XML data to store these example data types. We did however create the relationships between the tables, simulating a realistic use case.

Textures were stored as binary BLOB datatypes in SQLite. The entire pixel buffer is stored in one column, with dedicated columns for width, height, depth, etc. When importing an existing OpenFlight terrain tile, the textures are loaded into the texture table, and assigned a unique texture identifier (using the SQLite default primary key 'rowid'). Then as the materials associated with a mesh are imported, a many to many join table is used to tie the material to the texture. A similar mechanism is used to associate meshes with materials and nodes to meshes.

One of the issues of concern is that a relational database is not an ideal way to store a graph. To store a graph, a join table is used to track parents and children of each node. Traversing the tree either requires a series of queries to read the nodes in a sub-graph, or a bulk query to read all the edges in a graph at once. Because this bulk read could be excessively large in a massive database, a more efficient approach may be needed. Future approaches to this problem are discussed in the 'Next Steps' portion of this paper.

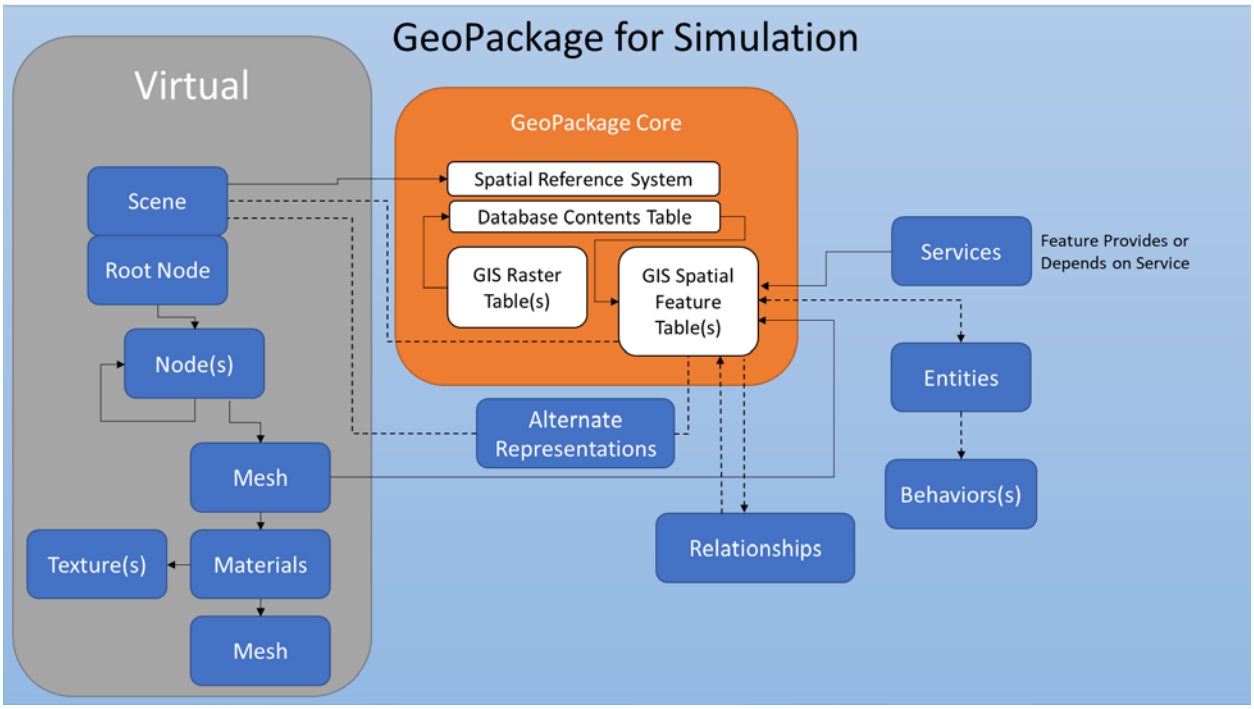

Figure 4 Schema Overview

Triggers

As mentioned previously in this paper, triggers in a relational database are automatic functions that are performed during specified events. Triggers can be used to enforce rules, such as preventing some data from being deleted except for certain operations, to create an audit trail when data is modified, or similar functions. SQLite does not support stored procedures as some database engines do, but it does allow some logic to be built into the database itself, which enforces behaviors regardless of the applications that use the data. This is one of the ways in which GeoPackage differs from most existing formats used in simulation.

Relationships

To allow relationships between features of all types, relationships are defined in a join table within the GeoPackage file as shown in Figure 6.[2] In this schema, we define relationships as having a child and a parent, but some relationships are symmetric (for example, two roads may have the relationship of 'intersects'). For this proof of concept, we treat all relationships as parent/child. Tables in a relationship are foreign keys to the existing GeoPackage gpkg_contents table. This table identifies the name of the table with the relationship. The row ids reference specific entries in the parent and child tables. The relationship type is enumerated in the secore_sim_relationship_types table.

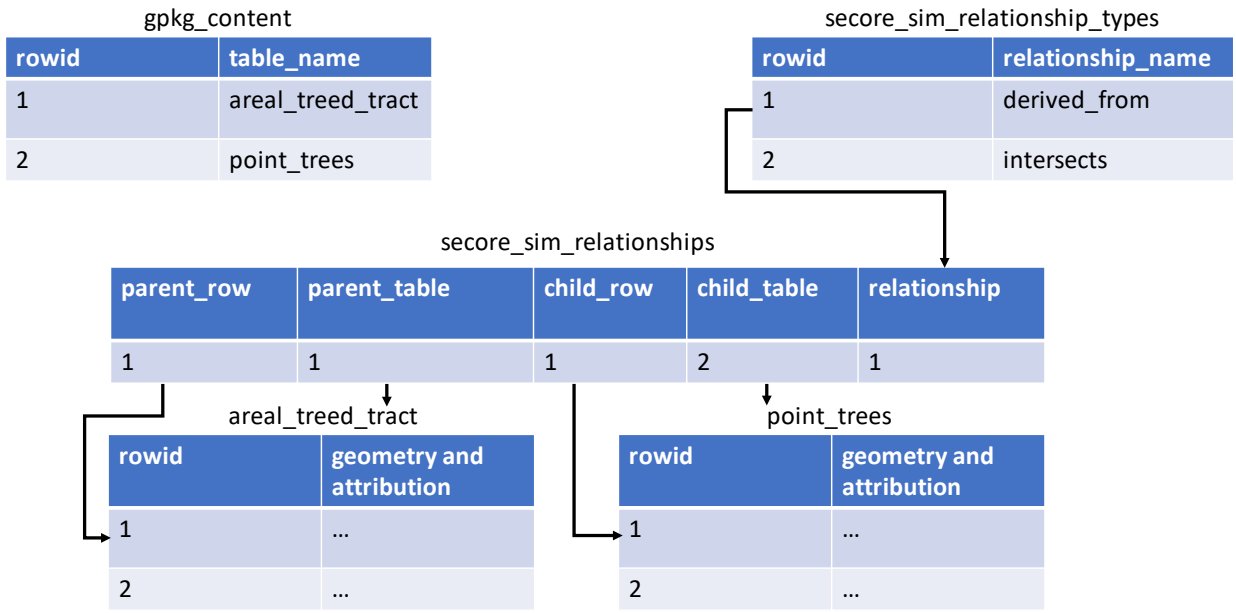

Figure 5 Relationships

Views

Views in a relational database such as SQLite act as a sort of virtual table by combining data across more than one table. For example, a view of services can associate all services with their corresponding GIS feature(s). This simplifies application development by making the associations automatically. For example, a GIS feature that provides a service could allow a spatial query for all features that provide a service of 'Electric Vehicle Charging' to be queried in the specified area. In this manner, a single GIS feature can provide multiple services that are easily found by an application, without duplication of spatial data. The benefit of this approach is that if the GIS feature is moved or adjusted, there is no need to update the corresponding services because they are already linked through a view.

It should be noted that views are read only, so while they provide a short-cut to the application developer for queries and data retrieval, the application must use the associated tables directly. Still, the relationships between tables is preserved when one feature is modified regardless of whether a view is used or not.

[2] There is an existing GeoPackage "Related Tables" extension that implements a similar concept for relationships. In our experiment we did not use the table and column naming convention, but for future development the existing extensions will be used.

RESULTS

Performance

Spatial indexes (these are implemented as R-Tree tables, not regular SQLite indexes) in a GeoPackage file were very effective in improving performance in large databases. Repeated, randomly generated spatial queries were performed over increasingly large datasets. Without spatial indexes, the time required to perform a query increased linearly with the number of features in the database. With the spatial index enabled, the time to perform a query remained nearly the same with each dataset. Without the spatial index, in our experiments, the average query across approximately 1,000,000 records took about six seconds, whereas with the spatial index, queries consistently took less than one millisecond.

Similarly, queries of non-spatial columns performed consistently better with appropriate indexes. Queries against non-indexed attributes slowed down approximately linearly with the number of records in the table, whereas queries against indexed columns remained relatively stable.

There is a cost associated with indexes including decreased performance when inserting or updating records, and increased file size for each index. In the case of the R-Tree spatial indexes, each feature that is indexed requires several columns of data in the associated R-Tree tables.

Knowledge about the kinds of queries that will be used is important in designing the indexes used for each table. The SQLite engine will pick indexes using its own rules, so knowledge about how the SQLite query planner works is important when adding indexes. Because SQLite (not the application) determines which index to use, in most cases applications do not need knowledge about which indexes exist.

GeoPackage geometries are not compressed, but stored as Well-Known-Binary (WKB). This can result in larger files than other formats that compress the data internally.

In our experiments, the performance of GeoPackage varied greatly not only by the design and use of indexes, but also tuning of the SQLite engine itself. The SQLite database allows the application to tune many parameters such as the cache page size and count as well as different journal modes. Different use cases (e.g. insert/update/read) may benefit from different settings. These settings are defined by the application that uses the GeoPackage file, not the file itself. Therefore, application developers should familiarize themselves with the SQLite options when developing GeoPackage software for maximum performance.

NEXT STEPS

Database Generation / Population

As previously described, GeoPackage provides a viable platform to unify mission command and simulation data. Current simulation database generation tools do not adequately preserve the relationships between these types of data. The relationships between data such as behaviors, entities, and services and the features in a GeoPackage database need to be established through either tools or human input. Ways in which existing tools can be used and/or enhanced, as well as opportunities for the development of new software should be explored.

An example of how a simulation enhanced GeoPackage could be used for database generation is creation of a module that would use triggers and relationships to automatically create and update derived data. For example, a GIS polygon feature for treed tract could automatically turn in to scattered external models. In this way, relationships between a GIS polygon, thepoints scattered from it, and the faces in the Triangle Irregular Network (TIN) would be preserved.

Most of the relationships in the data during our experiments were not readily viewed in a graphic environment due to existing software tools being focused on either simulation *or* mission command/GIS. As part of a further experiment, we recommend the enhancement of an existing tool (many tools are candidates for this enhancement) to navigate and visualize the layers of data and the relationships between them. For example, while displaying a 3D terrain environment, if a user clicks on a school building, all the related GIS features, alternate representation, and services data would be shown to the user, and graph of all the relationships within the database would be navigable.

A simulation enhanced GeoPackage has the potential to work as a repository for a very large simulation dataset. A test of the spatial indexes over a very large database could include collision and line of sight testing for areas outside an area that is currently paged in memory by using the database engine and indexes such as the built in R-Tree. GeoPackage databases in the Terabyte range should be populated and tested for performance.

Additional Data Representation

This experiment focused on geometry, textures, and relationships as well as limited support for constructive simulation. A full test/demonstration of a simulation enhanced GeoPackage would include support for additional bounding boxes, physics, materials, and more.

While we propose a single database that would include enough data to support a wide range of simulation systems including gaming, there are still vendor and system specific requirements in the data. With GeoPackage, there is the ability for a vendor or system to extend further by adding some self-describing data necessary for some systems. Additional experiments in this area would be useful.

Data Model

For our experiment, only a minimal ad-hoc schema and data model was created. The schema would need to be fully developed in coordination with the OGC to present a full proposal to the OGC technical committee. In addition to a proposed schema structured data structured data models for behaviors, services, entities (e.g. moving models), and other simulation specific data would need to be defined an accepted by the simulation community to provide a solution that will be widely adopted.

CONCLUSION/SUMMARY

In our experiments, GeoPackage was an effective foundation for transmittal and storage of mixed GIS and simulation content. Building a simulation extension for GeoPackage could improve interoperability, the opportunity for data sharing and re-use, while unifying and sharing mission command/GIS data with the data used in simulation programs. Additional R&D and collaboration between the mission command and simulation communities to refine and standardize extensions are needed to realize the benefits of the adoption of GeoPackage as a simulation format.

ACKNOWLEDGEMENTS

The authors acknowledge the PEOSTRI SE Core team as well as Leidos for contribution and support of this experiment and the development and success of this paper.

REFERENCES

35% Faster Than The Filesystem. (2017, 06 14). Retrieved from SQLite: https://www.sqlite.org/fasterthanfs.html

Dumanoir, P., Clinger, B., & Rivera, J. (2009). Evolving Standards in US Army Live Simulations. *Fall Simulation Interoperability Workshop 2009.* Orlando, Florida.

Freeman, J., & Srouji, B. (2017). Alternative Deployment Techniques for OGC CDB. *IMAGE*.

Layered Terrain Format ; Format Specification, Schema, and Layer Profiles. (2016, March 15).