Open Geospatial Consortium Inc.

Date: 2013-08-01

Reference number of this OGC® project document: OGC 12-128r8

Version: 0.8.0

Category: OGC® Implementation Specification

Editor: Paul Daisey

OpenGIS® GeoPackage Implementation Specification

Document type:	OGC® Publicly Available Standard
Document subtype:	Implementation Specification
Document stage:	Draft
Document language:	English

a)

Introduction

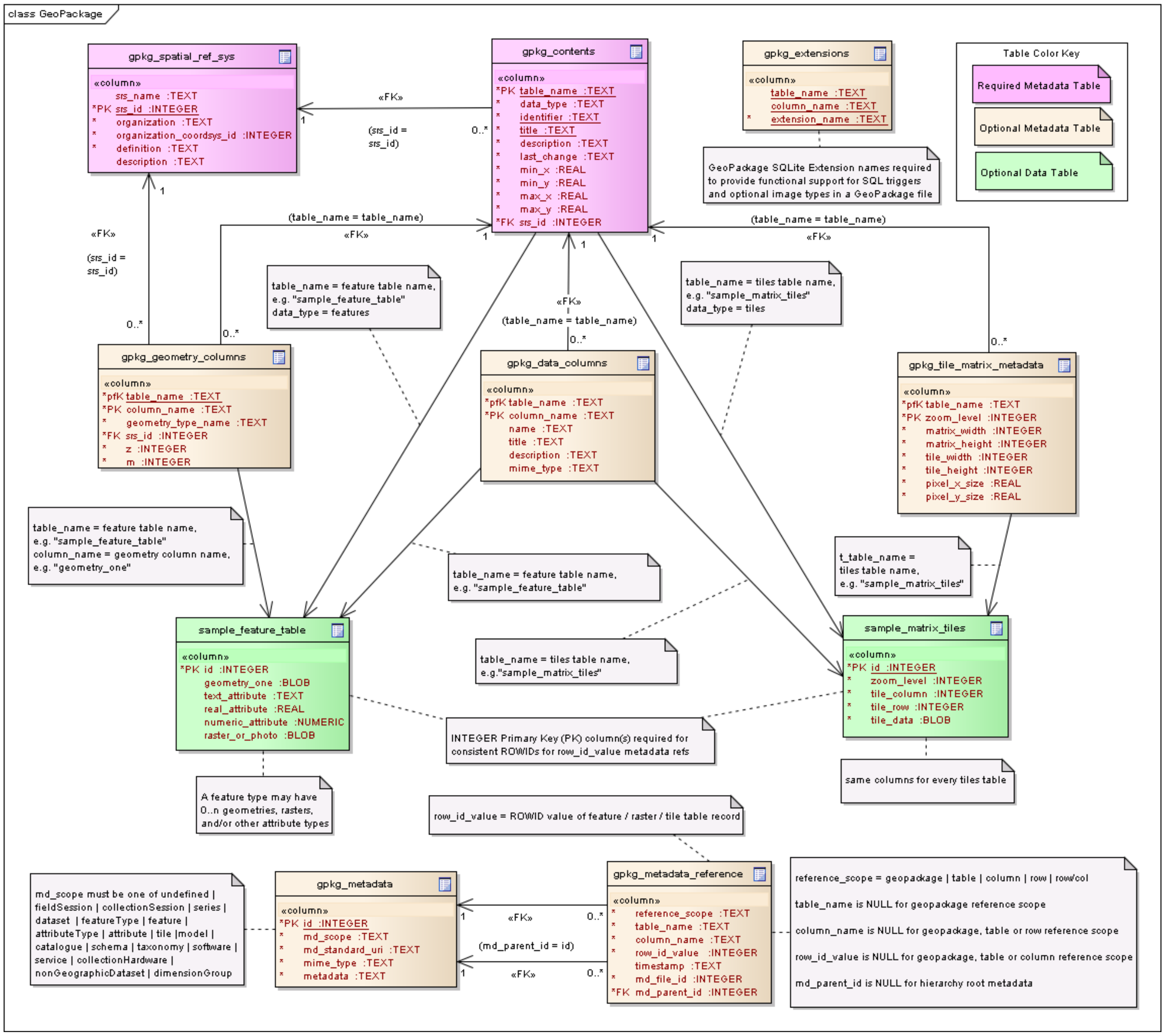
Mobile device users who require map/geospatial application services and operate in disconnected or limited network connectivity environments are challenged by limited storage capacity and the lack of open format geospatial data to support these applications. The current situation is that each map/geospatial application requires its own potentially proprietary geospatial data store. These separate application-specific data stores may contain the same geospatial data, wasting the limited storage available, and requiring custom applications for data translation, replication, and synchronization to enable different map/geospatial applications to share the same world view. In addition, many existing geospatial data stores are platform-specific, which means that users with different platforms must translate data to share it.

An open, standards-based, application-independent, platform-independent, portable, interoperable, self-describing, GeoPackage (GPKG) data container, API and manifest are needed to overcome these challenges and to effectively support multiple map/geospatial applications such as fixed product distribution, local data collection, and geospatially enabled analytics. This standard is intended to facilitate widespread adoption and use of GeoPackages by both COTS and open-source software applications on enterprise production platforms as well as mobile hand-held devices, given that mobile hand held devices do not yet have the processing power or battery life to effectively tackle difficult geospatial product production and analysis tasks. An application that accesses a GPKG will make use of the GPKG capabilities it requires; few if any such applications will make use of all GPKG capabilities.

This OpenGIS ® Implementation Specification defines GeoPackage files for exchange and GeoPackage SQLite Extensions for direct use of vector geospatial features and / or tile matrix sets of earth images and raster maps at various scales. Direct use means the ability to access and update data in a "native" format without intermediate format translations in an environment (e.g. through an API) that guarantees data model and data set integrity and identical access and update results in response to identical requests from different client applications.

A **GeoPackage file** is a platform-independent SQLite database file that contains GeoPackage data and metadata tables shown in Figure 1, with specified definitions, integrity assertions, format limitations and content constraints.

Figure 1- GeoPackage Tables


class GeoPackage
gpkg_spatial_ref_sys
«column»
srs_name :TEXT
*PK srs_id :INTEGER
* organization :TEXT
* organization_coordsys_id :INTEGER
* definition :TEXT
description :TEXT
gpkg_contents
«column»
*PK table_name :TEXT
* data_type :TEXT
* identifier :TEXT
* title :TEXT
* description :TEXT
* last_change :TEXT
* min_x :REAL
* min_y :REAL
* max_x :REAL
* max_y :REAL
*FK srs_id :INTEGER
gpkg_extensions
«column»
table_name :TEXT
column_name :TEXT
* extension_name :TEXT
Table Color Key
Required Metadata Table
Optional Metadata Table
Optional Data Table
GeoPackage SQLite Extension names required to provide functional support for SQL triggers and optional image types in a GeoPackage file
«FK»
(srs_id = srs_id)
1
0..*
«FK»
(srs_id = srs_id)
(table_name = table_name)
«FK»
(table_name = table_name)
«FK»
table_name = feature table name, e.g. "sample_feature_table" data_type = features
table_name = tiles table name, e.g. "sample_matrix_tiles" data_type = tiles
«FK»
(table_name = table_name)
gpkg_geometry_columns
«column»
*pfK table_name :TEXT
*PK column_name :TEXT
* geometry_type_name :TEXT
*FK srs_id :INTEGER
* z :INTEGER
* m :INTEGER
gpkg_data_columns
«column»
*pfK table_name :TEXT
*PK column_name :TEXT
name :TEXT
title :TEXT
description :TEXT
mime_type :TEXT
gpkg_tile_matrix_metadata
«column»
*pfK table_name :TEXT
*PK zoom_level :INTEGER
* matrix_width :INTEGER
* matrix_height :INTEGER
* tile_width :INTEGER
* tile_height :INTEGER
* pixel_x_size :REAL
* pixel_y_size :REAL
table_name = feature table name, e.g. "sample_feature_table" column_name = geometry column name, e.g. "geometry_one"
table_name = feature table name, e.g. "sample_feature_table"
t_table_name = tiles table name, e.g. "sample_matrix_tiles"
sample_feature_table
«column»
*PK id :INTEGER
geometry_one :BLOB
text_attribute :TEXT
real_attribute :REAL
numeric_attribute :NUMERIC
raster_or_photo :BLOB
table_name = tiles table name, e.g."sample_matrix_tiles"
sample_matrix_tiles
«column»
*PK id :INTEGER
* zoom_level :INTEGER
* tile_column :INTEGER
* tile_row :INTEGER
* tile_data :BLOB
INTEGER Primary Key (PK) column(s) required for consistent ROWIDs for row_id_value metadata refs
same columns for every tiles table
A feature type may have 0..n geometries, rasters, and/or other attribute types
row_id_value = ROWID value of feature / raster / tile table record
md_scope must be one of undefined | fieldSession | collectionSession | series | dataset | featureType | feature | attributeType | attribute | tile | model | catalogue | schema | taxonomy | software | service | collectionHardware | nonGeographicDataset | dimensionGroup
gpkg_metadata
«column»
*PK id :INTEGER
* md_scope :TEXT
* md_standard_uri :TEXT
* mime_type :TEXT
* metadata :TEXT
«FK»
(md_parent_id = id)
«FK»
gpkg_metadata_reference
«column»
* reference_scope :TEXT
* table_name :TEXT
* column_name :TEXT
* row_id_value :INTEGER
timestamp :TEXT
* md_file_id :INTEGER
*FK md_parent_id :INTEGER
reference_scope = geopackage | table | column | row | row/col
table_name is NULL for geopackage reference scope
column_name is NULL for geopackage, table or row reference scope
row_id_value is NULL for geopackage, table or column reference scope
md_parent_id is NULL for hierarchy root metadata

A **GeoPackage file** MAY be "empty" (contain user data table(s) for vector features 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 file metadata can describe GeoPackage data contents and identify external data synchronization sources and targets. A GeoPackage file MAY contain spatial indexes on feature geometries and SQL triggers to maintain indexes and enforce content constraints.

A **GeoPackage SQLite Extension** that MAY provide a minimal set of SQL API runtime functions to support spatial indexes and SQL triggers linked to a SQLite library with specified configuration requirements to provide SQL API access to a GeoPackage file. A **GeoPackage** is a **GeoPackage file** used with a **GeoPackage SQLite Extension.** A GeoPackage MAY have extensions that are registered in the gpkg_extensions table.

This specification does not address the issues listed in the B.6 Potential Future Work clause in Annex B, which MAY be addressed in a subsequent version of this specification or by other specifications.

OpenGIS® GeoPackage Implementation Specification

Base

The required capabilities specified in this clause serve as the base for optional options and registered extensions capabilities specified in clauses 1 and 2.

1.1 Core

The mandatory core capabilities defined in sub clauses and requirement statements of this clause SHALL be implemented by every **GeoPackage**.

1.1.1. SQLite Container

A self-contained, single-file, cross-platform, serverless, transactional, open source RDBMS container is desired to simplify production, distribution and use of GeoPackages and guarantee the integrity of the data they contain. “Self-contained” means that container software requires very minimal support from external libraries or from the operating system. “Single-file” means that a container not currently opened by any software application consists of a single file in a file system supported by a computing platform operating system. “Cross-platform” means that a container file MAY be created and loaded with data on one computing platform, and used and updated on another, even if they use different operating systems, file systems, and byte order (endian) conventions. “Serverless” means that the RDBMS container is implemented without any intermediary server process, and accessed directly by application software. “Transactional” means that RDBMS transactions guarantee that all changes to data in the container are Atomic, Consistent, Isolated, and Durable (ACID) despite program crashes, operating system crashes, and power failures.

1.1.1.1. Data

1.1.1.1.1. File Format

Req 1: ***The GeoPackage file format SHALL be a SQLite [9] database version 3 format [10][11] file, the first 16 bytes of which contain “SQLite format 3”[1] in ASCII.[2]***

It is RECOMMENDED that data providers whose data is intended for use in multiple applications put the value “GPKG” in the SQLite database header application id field. However, applications that put application-specific data in a GeoPackage MAY put their own application acronym in the application id field, so this is NOT a GeoPackage requirement.

The maximum size of a GeoPackage file is about 140TB. In practice a lower size limit MAY be imposed by the filesystem to which the file is written. Many mobile devices require external memory cards to be formatted using the FAT32 file system which imposes a maximum size limit of 4GB.

[1] SQLite version 4 [B49], which will be an alternative to version 3, not a replacement thereof,was not available when this specification was written. . See Future Work clause in Annex B.

[2] SQLite is in the public domain (see http://www.sqlite.org/copyright.html)

Field Cod

1.1.1.1.2. File Extension Name

Req 2: ***GeoPackage files SHALL have the file extension name ".gpkg".***

1.1.1.1.3. Integrity Check

Req 3: ***The SQLite PRAGMA integrity_check SQL command SHALL return "ok" for a GeoPackage file.***

1.1.1.2. API

1.1.1.2.1. Structured Query Language (SQL)

Req 4: ***A GeoPackage SQLite Extension SHALL provide SQL access to GeoPackage file contents via SQLite sqlite3 [12] software APIs.***[1]

1.1.1.2.2. Every GPKG SQLite Configuration

The SQLite [12] library has many compile time and run time options that MAY be used to configure SQLite for different uses. A GeoPackage application can use the compile options and other SQLite pragmas to get the effective compile and run time option settings and compare them to those required for a particular GeoPackage.

Req 5: ***Every GeoPackage SQLite Extension SHALL have the SQLite library compile and run time options specified in clause 1.1.1.2.2 table 1.***

Table 1 Every GeoPackage SQLite Configuration

Setting	Option	Shall / Not	Discussion
compile	SQLITE_OMIT_AUTOINCREMENT	Not	A number of tables in GeoPackage are specified to have autoincrement columns.
compile	SQLITE_OMIT_DATETIME_FUNCS	Not	gpkg_contents table last_change column requires timestamps
compile	SQLITE_OMIT_DEPRECATED	Shall	Omit PRAGMAs marked "Do not use this pragma!" in SQLite documentation.
compile	SQLITE_OMIT_FLOATING_POINT	Not	min/max x/y columns in gpkg_contents table are type REAL

[1] New applications should use the latest available SQLite version software [12]

compile	SQLITE_OMIT_PRAGMA	Not	GeoPackage configuration checks will require use of the compile_options pragma.
compile	SQLITE_OMIT_FLAG_PRAGMAS	Not	GeoPackage conformance validation will require use of the integrity_check pragma
compile	SQLITE_OMIT_VIEW	Not	GeoPackage tables MAY be implemented as updateable views.

1.1.2. Spatial Reference Systems

1.1.2.1. Data

1.1.2.1.1. Table Definition

Req 6: ***A GeoPackage file SHALL include a gpkg_spatial_ref_sys table or updateable view per clause 1.1.2.1.1, table 2 and Table 19.***

A table or updateable view named gpkg_spatial_ref_sys is the first component of the standard SQL schema for simple features described in clause 2.1.1 below. The coordinate reference system definitions it contains are referenced by the GeoPackage gpkg_contents and gpkg_geometry_columns tables to relate the vector and tile data in user tables to locations on the earth.

The gpkg_spatial_ref_sys table includes at a minimum the columns specified in SQL/MM (ISO 13249-3) [16] and shown in table 2 below containing data that defines spatial reference systems. This table or view MAY include additional columns to meet the requirements of implementation software or other specifications. Views of this table or view MAY be used to provide compatibility with the SQL/MM [16] (Table 20) and OGC Simple Features SQL [14][15] (Table 21) specifications.

Table 2 Spatial Ref Sys Table or View Definition

Column Name	**Column Type**	**Column Description**	**Key**
srs_name	text	Human readable name of this SRS	
srs_id	integer	Unique identifier for each Spatial Reference System within a GeoPackage file	PK
organization	text	Case-insensitive name of the defining organization e.g. EPSG or epsg	
organization_coordsys_id	integer	Numeric ID of the Spatial Reference System assigned by the organization	
definition	text	Well-known Text Representation of the Spatial Reference System	

Field Cod

description	text	Human readable description of this SRS	

See Annex C Table Definition SQL (Normative) C.1 gpkg_spatial_ref_sys.

1.1.2.1.2. Table Data Values

Req 7: ***The gpkg_spatial_ref_sys table or updateable view in a GeoPackage SHALL contain a record for organization EPSG or epsg and organization_coordsys_id 4326 [17][18] for WGS-84 [19], a record with an srs_id of -1, an organization of "NONE", an organization_coordsys_id of -1, and definition "undefined" for undefined Cartesian coordinate reference systems, and a record with an srs_id of 0, an organization of "NONE", an organization_coordsys_id of 0, and definition "undefined" for undefined geographic coordinate reference systems.***

Req 8: ***The spatial_ref_sys table or updateable view in a GeoPackage file SHALL contain records to define all spatial reference systems used by features and tiles in a GeoPackage.***

1.1.3. Contents

1.1.3.1. Data

1.1.3.1.1. Table Definition

Req 9: ***A GeoPackage file SHALL include a gpkg_contents table or updateable view per clause 1.1.3.1.1, table 3 and table 22.***

The purpose of the gpkg_contents table is to provide identifying and descriptive information that an application can display to a user in a menu of geospatial data that is available for access and/or update.

Table 3 Contents Table or View Definition

Column Name	**Type**	**Description**	**Null**	**Default**	**Key**
table_name	text	The name of the tiles, or feature table	no		PK
data_type	text	Type of data stored in the table:. "features" per clause 2.1.2.1.1, "tiles" per clause 2.2.2.1.1, or an implementer-defined value for other data tables per clause 2.5.	no		
identifier	text	A human-readable identifier (e.g. short name) for the table_name content	yes		
description	text	A human-readable description for the table_name content	yes	""	
last_change	text	timestamp value in ISO 8601format as defined by the strftime function '%Y-%m-%dT%H:%M:%fZ' format string	no	strftime('%Y-%m-%dT%H:%M:%fZ', CURRENT_TIMESTAMP)	

Field Cod

		applied to the current time			
min_x	double	Bounding box for all content in table_name	yes		
min_y	double	Bounding box for all content in table_name	yes		
max_x	double	Bounding box for all content in table_name	yes		
max_y	double	Bounding box for all content in table_name	yes		
srs_id	integer	Spatial Reference System ID: gpkg_spatial_ref_sys.srs_id; when data_type is features, SHALL also match gpkg_geometry_columns.srs_id	yes		FK

The gpkg_contents table is intended to provide a list of all geospatial contents in the GeoPackage. The data_type specifies the type of content. The bounding box (min_x, min_y, max_x, max_y) provides an informative bounding box (not necessarily minimum bounding box) of the content. If the srs_id column value references a geographic coordinate reference system (CRS), then the min/max x/y values are in decimal degrees; otherwise, the srs_id references a projected CRS and the min/max x/y values are in the units specified by that CRS.

See Annex C Table Definition SQL (Normative) C.2 gpkg_contents.

1.1.3.1.2. Table Data Values

Req 10:*The table_name column value in a gpkg_contents table row SHALL contain the name of a SQLite table or view.*

Req 11:*Values of the gpkg_contents table last_change column SHALL be in ISO 8601 [41] format containing a complete date plus UTC hours, minutes, seconds and a decimal fraction of a second, with a 'Z' ('zulu') suffix indicating UTC.*[1]

Req 12:*Values of the gpkg_contents table srs_id column SHALL reference values in the spatial_ref_sys table srs_id column.*

Options

The optional capabilities specified in this clause depend on the required capabilities specified in 1 Base above..

Req 13:*A Valid GeoPackage SHALL contain features per clause 2.1 and/or tiles per clause 2.2 and row(s) in the gpkg_contents table with data_type column values of "features" and/or "tiles" describing the user data tables.*

[1] The following statement selects an ISO 8601timestamp value using the SQLite strftime function:
```
SELECT (strftime('%Y-%m-%dT%H:%M:%fZ','now')).
```

Field Cod

2.1. Features

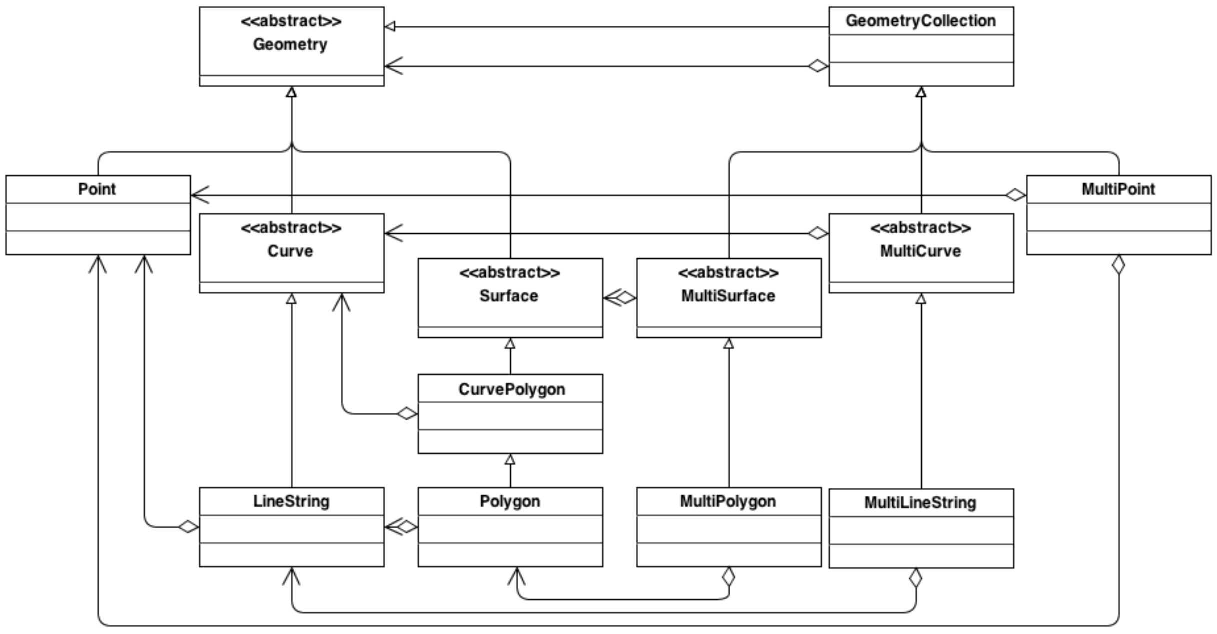
The optional capabilities defined in sub clauses and requirement statements of this clause MAY be implemented by any **GeoPackage file.**

2.1.1. Simple Features SQL Introduction

Vector feature data represents geolocated entities including conceptual ones such as districts, real world objects such as roads and rivers, and observations thereof. International specifications [14][15][16] have standardized practices for the storage, access and use of vector geospatial features and geometries via SQL in relational databases. The first component of the SQL schema for vector features in a GeoPackage is the gpkg_spatial_ref_sys table defined in clause 1.1.2 above. Other components are defined below.

In a GeoPackage file, "simple" features are geolocated using a linear geometry subset of the SQL/MM (ISO 13249-3) [16] geometry model shown in figure 2 below.

Figure 2 – Core Geometry Model



The instantiable (not abstract) geometry types defined in this Standard are restricted to 0, 1 and 2-dimensional geometric objects that exist in 2, 3 or 4-dimensional coordinate space (R2, R3 or R4). Geometry values in R2 have points with coordinate values for x and y. Geometry values in R3 have points with coordinate values for x, y and z or for x, y and m. Geometry values in R4 have points with coordinate values for x, y, z and m. The interpretation of the coordinates is subject to the coordinate reference systems associated to the point. All coordinates within a geometry object should be in the same coordinate reference systems.

Field Cod

Geometries may include z coordinate values. The z coordinate value traditionally represents the third dimension (i.e. 3D). In a Geographic Information System (GIS) this may be height above or below sea level. For example: A map might have a point identifying the position of a mountain peak by its location on the earth, with the x and y coordinate values, and the height of the mountain, with the z coordinate value.

Geometries may include m coordinate values. The m coordinate value allows the application environment to associate some measure with the point values. For example: A stream network may be modeled as multilinestring value with the m coordinate values measuring the distance from the mouth of stream.

All geometry types described in this standard are defined so that instances of Geometry are topologically closed, i.e. all represented geometries include their boundary as point sets. This does not affect their representation, and open version of the same classes may be used in other circumstances, such as topological representations.

A brief description of each geometry type is provided below. A more detailed description can be found in ISO 13249-3[16].

- Geometry: the root of the geometry type hierarchy.
- Point: a single location in space. Each point has an X and Y coordinate. A point may optionally also have a Z and/or an M value.
- Curve: the base type for all 1-dimensional geometry types. A 1-dimensional geometry is a geometry that has a length, but no area. A curve is considered simple if it does not intersect itself (except at the start and end point). A curve is considered closed its start and end point are coincident. A simple, closed curve is called a ring.
- LineString: A Curve that connects two or points in space.
- Surface: the base type for all 2-dimensional geometry types. A 2-dimensional geometry is a geometry that has an area.
- CurvePolygon: A planar surface defined by an exterior ring and zero or more interior ring. Each ring is defined by a Curve instance.
- Polygon: A restricted form of CurvePolygon where each ring is defined as a simple, closed LineString.
- GeometryCollection: A collection of zero or more Geometry instances.
- MultiSurface: A restricted form of GeometryCollection where each Geometry in the collection must be of type Surface.
- MultiPolygon: A restricted form of MultiSurface where each Surface in the collection must be of type Polygon.
- MultiCurve: A restricted form of GeometryCollection where each Geometry in the collection must be of type Curve.
- MultiLineString? : A restricted form of MultiCurve where each Curve in the collection must be of type LineString.
- MultiPoint: A restricted form of GeometryCollection where each Geometry in the collection must be of type Point.

Field Cod

2.1.2. Contents

2.1.2.1. Data

2.1.2.1.1. Contents Table – Features Row

Req 14: ***gpkg_contents SHALL contain a row with a data_type column value of "features" for each vector features user data table.***

2.1.3. Geometry Encoding

2.1.3.1. Data

2.1.3.1.1. BLOB Format

Req 15: ***A GeoPackage file SHALL store feature table geometries with or without optional elevation (Z) and/or measure (M) values in SQL BLOBs using the GeoPackageBinary format specified in table 4 and clause 2.1.3.1.1.***

Table 4 GeoPackage SQL Geometry Binary Format

```
GeoPackageBinary {
  byte[2] magic = 0x4750; // 'GP'
  byte version;           // 8-bit unsigned integer, 0 = version 1
  byte flags;             // see flags layout below
  int32 srs_id;
  double[] envelope;      // see flags envelope contents indicator code below
  WKBGeometry geometry;   // per  ISO 13249-3[16] clause 5.1.46 [1]
}
```

flags layout:

bit	7	6	5	4	3	2	1	0
use	R	R	R	Y	E	E	E	B

use:

R: reserved for future use; set to 0

Y: empty geometry flag
   0: non-empty geometry
   1: empty geometry

E: envelope contents indicator code (3-bit unsigned integer)

code	**description**	**envelope**

[1] OGC WKB simple feature geometry types specified in [13] are a subset of the ISO WKB geometry types specified in [16]

Field Cod

value		length (bytes)
0	no envelope (space saving slower indexing option)	0
1	envelope is [minx, maxx, miny, maxy]	32
2	envelope is [minx, maxx, miny, maxy, minz, maxz]	48
3	envelope is [minx, maxx, miny, maxy, minm, maxm]	48
4	envelope is [minx, maxx, miny, maxy, minz, maxz, minm, maxm]	64
5-7	invalid	unknown

B: byte order for header values (1-bit Boolean)
 0 = Big Endian (most significant byte first)
 1 = Little Endian (least significant byte first)

Well-Known Binary as defined in ISO 13249-3 [16] does not provide a standardized encoding for an empty point set (i.e., 'Point Empty' in Well-Known Text). In GeoPackage files these points SHALL be encoded as a Point where each coordinate value is set to an IEEE-754 quiet NaN value. GeoPackage files SHALL use big endian 0x7ff8000000000000 or little endian 0x000000000000f87f as the binary encoding of the NaN values.

When the WKBGeometry in a GeoPackageBinary is empty, either the envelope contents indicator code SHALL be 0 indicating no envelope, or the envelope SHALL have its values set to NaN as defined for an empty point.

2.1.3.2. API

2.1.3.2.1. Minimal Runtime SQL Functions

In contrast to functions in application code or a runtime library, triggers are part of the SQLite database file. When an application writes to a GeoPackage file that it did not create itself then there is the possibility that it will invoke a trigger that calls a function that the application's runtime library does not provide. To avoid this interoperability problem, a small set of functions on the GeoPackageBinary geometry specified in clause 2.1.3.1.1 are defined in Annex D. Every implementation can be sure that triggers that only use these functions in addition to those provided by SQLite will work as intended across implementations. [1][2]

Req 16:*A GeoPackage SQLite Extension MAY provide SQL function support for triggers in GeoPackage file. One that does so SHALL provide the minimal runtime SQL functions listed in Annex D table 36.*

[1] Functions other than the minimal runtime SQL functions required by triggers in a GeoPackage file SHOULD be documented in the gpkg_extensions table and provided by a GeoPackage SQLite Extension.

[2] SQL functions on geometries in addition to those defined in this clause should conform to the SF/SQL [14][15] and SQL/MM [16] specifications cited in clause 2.1.1 above.

Field Cod

2.1.4. Geometry Types

2.1.4.1. Data

2.1.4.1.1. Core Types

Req 17:*A GeoPackage file SHALL store feature table geometries with the basic simple feature geometry types (Geometry, Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon, GeomCollection) in Annex G table 46 in the GeoPackageBinary geometry encoding format.*

2.1.5. Geometry Columns

2.1.5.1. Data

2.1.5.1.1. Table Definition

Req 18:*A GeoPackage file with a gpkg_contents table row with a "features" data_type SHALL contain a gpkg_geometry_columns table or updateable view per clause 2.1.5.1.1, table 5 and Table 24.*

The second component of the SQL schema for vector features in a GeoPackage is a gpkg_geometry_columns table that identifies the geometry columns in tables that contain user data representing features. This table or updateable view SHALL contain one row record for each geometry column in each vector feature data table (clause 2.1.6) in a GeoPackage.

Table 5 Geometry Columns Table or View Definition

Column Name	**Type**	**Description**	**Key**
table_name	text	Name of the table containing the geometry column	PK, FK
column_name	text	Name of a column in the feature table that is a Geometry Column	PK
geometry_type_name	text	Name from table 46 or table 47 in Annex G	
srs_id	integer	Spatial Reference System ID: gpkg_spatial_ref_sys.srs_id	FK
z	integer	0: z values prohibited 1: z values mandatory 2: z values optional	
m	integer	0: m values prohibited 1: m values mandatory 2: m values optional	

The FK on gpkg_geometry_columns.srs_id references the PK on gpkg_spatial_ref_sys.srs_id to ensure that geometry columns are only defined in feature tables for defined spatial reference systems.

Field Cod

The gpkg_geometry_columns table or view MAY include additional columns to meet the requirements of implementation software or other specifications. Views of this table or view MAY be used to provide compatibility with the SQL/MM [16] (Table 25) and OGC Simple Features SQL [14][15] (Table 26) specifications.

See clause C.4 gpkg_geometry_columns.

2.1.5.1.2. Table Data Values

Req 19:*Values of the gpkg_geometry_columns table table_name column SHALL reference values in the gpkg_contents table_name column.*

Req 20:*The column_name column value in a gpkg_geometry_columns table row SHALL be the name of a column in the table specified by the table_name column value for that row.*

Req 21:*The geometry_type_name value in a gpkg_geometry_columns table row SHALL be one of the geometry type names specified in Annex G.*

Req 22:*The srs_id value in a gpkg_geometry_columns table row SHALL be an srs_id column value from the gpkg_spatial_ref_sys table.*

Req 23:*The z value in a gpkg_geometry_columns table row SHALL be one of 0, 1, or 2.*

Req 24:*The m value in a gpkg_geometry_columns table row SHALL be one of 0, 1, or 2.*

2.1.6. Vector Feature User Data Tables

2.1.6.1. Data

2.1.6.1.1. Table Definition

The third component of the SQL schema for vector features in a GeoPackage described in clause 2.1.1 above are tables that contain user data representing features. Feature attributes are columns in a feature table, including geometries.[1] Features are rows in a feature table.[2]

Req 25:*A GeoPackage file MAY contain tables or updateable views containing vector features. Every such feature table or view in a GeoPackage file SHALL have a primary key defined on one integer column per table 6 and table 27.*

The integer primary key of a feature table allows features to be linked to row level metadata records in the gpkg_metadata table by rowid values in the gpkg_metadata_reference table as described in clause 2.4.3 below.

[1] A feature type MAY be defined to have 0..n geometry and/or raster attributes, so the corresponding feature table MAY contain from 0..n geometry and/or raster columns.

[2] A GeoPackage is not required to contain any feature data tables. Feature data tables in a GeoPackage MAY be empty

Field Cod

Req 26:*A feature table SHALL have only one geometry column.*

Feature data models from non-GeoPackage implementations that have multiple geometry columns per feature table MAY be transformed into GeoPackage? implementations with a separate feature table for each geometry type whose rows have matching integer primary key values that allow them to be joined in a view with the same column definitions as the non-GeoPackage feature data model with multiple geometry columns.

Table 6 EXAMPLE: Sample Feature Table or View Definition

Column Name	Type	Description	Null	Default	Key
id	integer	Autoincrement primary key	no		PK
geometry_one	BLOB	GeoPackage Geometry	no		
text_attribute	text	Text attribute of feature	no		
real_attribute	real	Real attribute of feature	no		
numeric_attribute	numeric	Numeric attribute of feature	no		
raster_or_ photo	BLOB	Photograph of the area	no		

See Annex C: Table Definition SQL clause C.5 sample_feature_table

2.1.6.1.2. Table Data Values

A feature geometry is stored in a geometry column specified by the geometry_column value for the feature table in the gpkg_geometry_columns table defined in clause 2.1.5 above.

The geometry type of a feature geometry column specified in the gpkg_geometry_columns table geometry_type_name column is a name from Annex G.

Req 27:*Feature table geometry columns SHALL contain geometries of the type or assignable for the type specified for the column by the gpkg_geometry_columns table geometry_type_name column value[1].*

Geometry subtypes are assignable as defined in Annex G and shown in part in Figure 2 – Core Geometry Model. For example, if the geometry_type_name value in the gpkg_geometry_columns table is for a geometry type like POINT that has no subtypes, then the feature table geometry column MAY only contain geometries of that type. If the geometry type_name value in the gpkg_geometry_columns table is for a geometry type like GEOMCOLLECTION that has subtypes, then the feature table geometry column MAY only contain geometries of that type or any of its direct or indirect subtypes. If the geometry type_name is GEOMETRY (the root of the geometry type hierarchy) then the feature table geometry column MAY contain geometries of any geometry type.

[1] GeoPackage applications MAY use SQL triggers or tests in application code to meet Req 27:

Field Cod

The presence or absence of optional elevation (Z) and/or measure (M) values in a geometry does not change its type or assignability.

The spatial reference system type of a feature geometry column specified by a gpkg_geometry_columns table srs_id column value is a code from the gpkg_spatial_ref_sys table srs_id column.

Req 28:*Feature table geometry columns SHALL contain geometries with the srs_id specified for the column by the gpkg_geometry_columns table srs_id column value.*

2.2. Tiles

The optional capabilities defined in sub-clauses and requirement statements of this clause MAY be implemented by any GeoPackage file.

2.2.1. Tile Matrix Introduction

There are a wide variety of commercial and open source conventions for storing, indexing, accessing and describing tiles in tile matrix pyramids. Unfortunately, no applicable existing consensus, national or international specifications have standardized practices in this domain. In addition, various image file formats have different representational capabilities, and include different self-descriptive metadata.

The tile store data / metadata model and conventions described below support direct use of tiles in a GeoPackage in two ways. First, they specify how existing applications MAY create SQL Views of the data /metadata model on top of existing application tables that that follow different interface conventions. Second, they include and expose enough metadata information at both the dataset and record levels to allow applications that use GeoPackage data to discover its characteristics without having to parse all of the stored images. Applications that store GeoPackage tile data, which are presumed to have this information available, SHALL store sufficient metadata to enable its intended use.

The GeoPackage tile store data model MAY be implemented directly as SQL tables in a SQLite database for maximum performance, or as SQL views on top of tables in an existing SQLite tile store for maximum adaptability and loose coupling to enable widespread implementation.

A GeoPackage CAN store multiple raster and tile pyramid data sets in different tables or views in the same container. [1]

The tables or views that implement the GeoPackage tile store data / metadata model are described and discussed individually in the following subsections.

[1] Images of multiple MIME types MAY be stored in given table. For example, in a tiles table, image/png format tiles without compression could be used for transparency where there is no data on the tile edges, and image/jpeg format tiles with compression could be used for storage efficiency where there is image data for all pixels. Images of multiple bit depths of the same MIME type MAY also be stored in a given table, for example image/png tiles in both 8 and 24 bit depths.

Field Cod

2.2.2. Contents

2.2.2.1 Data

2.2.2.1.1. Contents Table – Tiles Row

Req 29:*A gpkg_contents table or updateable view SHALL contain a row with a data_type column value of "tiles" for each tile matrix user data table.*

2.2.3. Zoom Levels

2.2.3.1. Data

2.2.3.1.1. Zoom Times Two

Req 30:*In a GeoPackage file that contains a tile matrix user data table that contains tile data, by default*[1]*, zoom level pixel sizes for that table SHALL vary by powers of 2 between adjacent zoom levels in the tile matrix metadata table.*

2.2.4. Tile Encoding PNG

2.2.4.1. Data

2.2.4.1.1. MIME Type PNG

Req 31:*In a GeoPackage file that contains a tile matrix user data table that contains tile data that is not MIME type image/jpeg[21][22][23], by default SHALL store that tile data in MIME type image/png [24][25].*[2]

2.2.5. Tile Encoding JPEG

2.2.5.1. Data

2.2.5.1.1. MIME Type JPEG

Req 32:*In a GeoPackage file that contains a tile matrix user data table that contains tile data that is not MIME type image/png [24][25], by default SHALL store that tile data in MIME type image/jpeg[21][22][23],* [3]

2.2.6. Tile Matrix Metadata

2.2.6.1. Data

2.2.6.1.1. Table Definition

Req 33:*A GeoPackage that contains a tile matrix user data table SHALL contain a gpkg_tile_matrix_metadata table per clause 2.2.6.1.1, Table 7 and Table 28.*

[1] See clause 3.2.1.1.1 for use of other zoom levels as a registered extensions.

[2] See Clauses 3.2.2, 3.2.3, 3.2.4 and 3.2.5 regarding use of alternative tile MIME types as registered extensions.

[3] See Clauses 3.2.2, 3.2.3, 3.2.4 and 3.2.5 regarding use of alternative tile MIME types as registered extensions.

Field Cod

Table 7 Tile Matrix Metadata Table or View Definition

Column Name	Column Type	Column Description	Null	Default	Key
table_name	text	{RasterLayerName}_tiles	no		PK, FK
zoom_level	integer	0 <= zoom_level <= max_level for t_table_name	no	0	PK
matrix_width	integer	Number of columns (>= 1) in tile matrix at this zoom level	no	1	
matrix_height	integer	Number of rows (>= 1) in tile matrix at this zoom level	no	1	
tile_width	integer	Tile width in pixels (>= 1)for this zoom level	no	256	
tile_height	integer	Tile height in pixels (>= 1) for this zoom level	no	256	
pixel_x_size	double	In t_table_name srs_id units or default meters for srs_id 0 (>0)	no	1	
pixel_y_size	double	In t_table_name srs_id units or default meters for srs_id 0 (>0)	no	1	

The gpkg_tile_matrix_metadata table documents the structure of the tile matrix at each zoom level in each tiles table. It allows GeoPackages to contain rectangular as well as square tiles (e.g. for better representation of polar regions). It allows tile pyramids with zoom levels that differ in resolution by powers of 2, irregular intervals, or regular intervals other than powers of 2.

See Annex C: Table Definition SQL clause C.6 gpkg_tile_matrix_metadata

2.2.6.1.2. Table Data Values

Req 34:*Values of the tile_matrix_metadata table_name column SHALL reference values in the gpkg_contents table_name column for rows with a data_type of “tiles”.*

Req 35:*A gpkg_tile_matrix_metadata table SHALL contain one row record for each zoom level that contains one or more tiles in each tile matrix user data table.*

The gpkg_tile_matrix_metadata table or view MAY contain row records for zoom levels in a tiles table that do not contain tiles.

GeoPackages follow the most frequently used conventions of a tile origin at the upper left and a zoom-out-level of 0 for the smallest map scale “whole world” zoom level view[1], as

[1] GeoPackage applications MAY query the gpkg_tile_matrix_metadata table or the tile matrix user data table to determine the minimum and maximum zoom levels for a given tile matrix table.

Field Cod

specified by WMTS [20]. The tile coordinate (0,0) always refers to the tile in the upper left corner of the tile matrix at any zoom level, regardless of the actual availability of that tile.

Req 36:*The zoom_level column value in a tile_matrix_metadata table row SHALL not be negative.*

Req 37:*The matrix_width column value in a tile_matrix_metadata table row SHALL be greater than 0.*

Req 38:*The matrix_height column value in a tile_matrix_metadata table row SHALL be greater than 0.*

Req 39:*The tile_width column value in a tile_matrix_metadata table row SHALL be greater than 0.*

Req 40:*The tile_height column value in a tile_matrix_metadata table row SHALL be greater than 0.*

Req 41:*The pixel_x_size column value in a tile_matrix_metadata table row SHALL be greater than 0.*

Req 42:*The pixel_y_size column value in a tile_matrix_metadata table row SHALL be greater than 0.*

Req 43:*The pixel_x_size and pixel_y_size column values for zoom_level column values in a tile_matrix_metadata table sorted in ascending order SHALL be sorted in descending order.*

Tiles MAY or MAY NOT be provided for level 0 or any other particular zoom level.[1] This means that a tile matrix set can be sparse, i.e. not contain a tile for any particular position at a certain tile zoom level.[2] This does not affect the spatial extent stated by the min/max x/y columns values in the gpkg_contents record for the same table_name, or the tile matrix width and height at that level.[3]

[1] GeoPackage applications MAY query the tiles (matrix set) table to determine which tiles are available at each zoom level.

[2] GeoPackage applications that insert, update, or delete tiles (matrix set) table tiles row records are responsible for maintaining the corresponding descriptive contents of the gpkg_tile_matrix_metadata table.

[3] The gpkg_contents table contains coordinates that define a bounding box as the stated spatial extent for all tiles in a tile (matrix set) table. If the geographic extent of the image data contained in these tiles is within but not equal to this bounding box, then the non-image area of matrix edge tiles must be padded with no-data values, preferably transparent ones.

Field Cod

2.2.7. Tile Matrix User Data

2.2.7.1. Data

2.2.7.1.1. Table Definition

Req 44:***Each tile matrix set in a GeoPackage file SHALL be stored in a different tiles table or updateable view with a unique name per clause 2.2.7.1.1, Table 8and Table 31.***

Table 8 Tiles Table or View Definition

Column Name	Column Type	Column Description	Null	Default	Key
id	integer	Autoincrement primary key	no		PK
zoom_level	integer	min(zoom_level) <= zoom_level <= max(zoom_level) for t_table_name	no	0	UK
tile_column	integer	0 to gpkg_tile_matrix_metadata matrix_width – 1	no	0	UK
tile_row	integer	0 to gpkg_tile_matrix_metadata matrix_height - 1	no	0	UK
tile_data	BLOB	Of an image MIME type specified in clauses 2.2.4, 2.2.5, 3.2.2, 3.2.3, 3.2.4	no		

See Annex C: Table **Definition SQL clause** C.7 sample_matrix_tiles

2.2.7.1.2. Table Data Values

Each tiles table or view[1] MAY contain tile matrices at zero or more zoom levels of different spatial resolution (map scale).

Req 45:***For each distinct table_name from the gpkg_tile_matrix_metadata (tmm) table, the tile matrix set (tms) user data table zoom_level column value in a GeoPackage file SHALL be in the range min(tmm.zoom_level) <= tms.zoom_level <= max(tmm.zoom_level).***

Req 46:***For each distinct table_name from the tile_matrix_metadata (tmm) table, the tile matrix set (tms) user data table tile_column column value in a GeoPackage file SHALL be in the range 0 <= tms.tile_column <= tmm.matrix_width – 1 where the tmm and tms zoom_level column values are equal.***

Req 47:***For each distinct table_name from the tile_matrix_metadata (tmm) table, the tile matrix set (tms) user data table tile_row column value in a GeoPackage file SHALL be in the range 0 <= tms.tile_row <= tmm.matrix_height – 1 where the tmm and tms zoom_level column values are equal.***

All tiles at a particular zoom level have the same pixel_x_size and pixel_y_size values specified in the gpkg_tile_matrix_metadata row record for that tiles table and zoom level.[1]

[1] A GeoPackage is not required to contain any tile matrix data tables. Tile matrix user data tables in a GeoPackage MAY be empty.

Field Cod

2.3. Schema

The optional capabilities defined in sub clauses and requirement statements of this clause MAY be implemented by any **GeoPackage file.**

2.3.1. Data Columns

2.3.1.1. Data

2.3.1.1.1. Table Definition

Req 48:*A GeoPackage file MAY contain a table or updateable view named gpkg_data_columns. If present it SHALL be defined per clause 2.3.1.1.1, Table 9 and Table 32.*

Table 9 Data Columns Table or View Definition

Column Name	**Column Type**	**Column Description**	**Null**	**Key**
table_name	text	Name of the tiles or feature table	no	PK
column_name	text	Name of the table column	no	PK
name	text	A human-readable identifier (e.g. short name) for the column_name content	yes	
title	text	A human-readable formal title for the column_name content	yes	
description	text	A human-readable description for the table_name content	yes	
mime_type	text	MIME [25] type of column_name if BLOB type, or NULL for other types	yes	

GeoPackage applications MAY[2] use the gpkg_data_columns table to store minimal application schema identifying, descriptive and MIME [25] type[3] information about columns in user vector feature and tile matrix data tables that supplements the data available from the SQLite sqlite_master table and pragma table_info(table_name) SQL function. The gpkg_data_columns data CAN be used to provide more specific column data types and value ranges and application specific structural and semantic information to enable more informative user menu displays and more effective user decisions on the suitability of GeoPackage contents for specific purposes.

[1] The zoom_level / tile_column / tile_row unique key is automatically indexed, and allows tiles to be selected and accessed by "z, x, y", a common convention used by some implementations. This table / view definition MAY also allow tiles to be selected based on a spatially indexed bounding box in a separate metadata table.

[2] A GeoPackage is not required to contain a gpkg_data_columns table. The gpkg_data_columns table in a GeoPackage MAY be empty.

[3] GeoPackages MAY contain MIME types other than the raster image types specified in clauses 2.2.4, 2.2.5, 3.2.2, 3.2.3, and 3.2.4 as feature attributes, but they are not required to do so

Field Cod

See Annex C: Table Definition SQL clause C.8 gpkg_data_columns

2.3.1.1.2. Table Data Values

Req 49:*Values of the gpkg_data_columns table table_name column value SHALL reference values in the gpkg_contents table_name column.*

Req 50:*The column_name column value in a gpkg_data_columns table row SHALL contain the name of a column in the SQLite table or view identified by the table_name column value.*

2.4. Metadata

The optional capabilities defined in sub clauses and requirement statements of this clause MAY be implemented by any **GeoPackage file.**

2.4.1. Introduction

Two tables in a GeoPackage file provide a means of storing metadata in MIME [25] encodings that are defined in accordance with any authoritative metadata specifications, and relating it to the features, rasters, and tiles data in a GeoPackage. These tables are intended to provide the support necessary to implement the hierarchical metadata model defined in ISO 19115 [40], Annex B B.5.25 MD_ScopeCode, Annex I and Annex J, so that as GeoPackage data is captured and updated, the most local and specific detailed metadata changes associated with the new or modified data MAY be captured separately, and referenced to existing global and general metadata.

The gpkg_metadata table that contains metadata is described in clause 2.4.2, and the gpkg_metadata_reference table that relates gpkg_metadata to GeoPackage data is described in clause 2.4.3. There is no GeoPackage requirement that such metadata be provided or that defined metadata be structured in a hierarchical fashion[1] with more than one level, only that if it is, these tables SHALL be used. Such metadata[2] and data that relates it to GeoPackage contents SHALL NOT be stored in other tables as defined in clause 2.5.

2.4.2. Metadata Table

2.4.2.1. Data

2.4.2.1.1. Table Definition

Req 51:*A GeoPackage file MAY contain a table or updateable view named gpkg_metadata. If present it SHALL be defined per clause2.4.2.1.1, Table 10and Table 33.*

The first component of GeoPackage metadata is the gpkg_metadata table that MAY contain metadata in MIME [25] encodings structured in accordance with any authoritative metadata specification, such as ISO 19115 [40], ISO 19115-2 [B31], ISO 19139 [B32], Dublin Core [B33], CSDGM [B34], DDMS [B35], NMF/NMIS [B36], etc. The GeoPackage interpretation

[1]: Informative examples of hierarchical metadata are provided in Annex I.

[2] An informative example of raster image metadata is provided in Annex J

Field Cod

of what constitutes “metadata” is a broad one that includes UML models [B37] encoded in XMI [B38], GML Application Schemas [B39], ISO 19110 feature catalogues [B40], OWL [B41] and SKOS [B42] taxonomies, etc.

Table 10 Metadata Table or View Definition

Column Name	Column Type	Column Description	Null	Default	Key
id	integer	Metadata primary key	no		PK
md_scope	text	Name of the data scope to which this metadata applies; see table 11 below	no	‘dataset’	
md_standard_uri	text	URI reference to the metadata structure definition authority	no	http://schemas.opengis.net/iso/19139/	
mime_type	text	MIME [25] encoding of metadata	no	text/xml	
metadata	text	metadata	no	’’	

See Annex C: Table Definition SQL clause C.9 gpkg_metadata.

2.4.2.1.2. Table Data Values

The md_scope column in the gpkg_metadata table is the name of the applicable scope for the contents of the metadata column for a given row. The list of valid scope names and their definitions is provided in Table 11below. The initial contents of this table were obtained from the ISO 19115 [40], Annex B B.5.25 MD_ScopeCode code list, which was extended[1] for use in the GeoPackage specification by addition of entries with “NA” as the scope code column in Table 10.

Table 11 Metadata Scopes

Name (md_scope)	Scope Code	Definition
undefined	NA	Metadata information scope is undefined
fieldSession	012	Information applies to the field session
collectionSession	004	Information applies to the collection session
series	006	Information applies to the (dataset) series[2]
dataset	005	Information applies to the (geographic feature) dataset
featureType	010	Information applies to a feature type (class)
feature	009	Information applies to a feature (instance)
attributeType	002	Information applies to the attribute class
attribute	001	Information applies to the characteristic of a feature (instance)

[1] The scope codes in Table 13 include a very wide set of descriptive information types as “metadata” to describe data.

[2] ISO 19139 format metadata [B32] is recommended for general-purpose description of geospatial data at the series and dataset metadata scopes.

Field Cod

tile	016	Information applies to a tile, a spatial subset of geographic data
model	015	Information applies to a copy or imitation of an existing or hypothetical object
catalog	NA	Metadata applies to a feature catalog[1]
schema	NA	Metadata applies to an application schema[2]
taxonomy	NA	Metadata applies to a taxonomy or knowledge system[3]
software	013	Information applies to a computer program or routine
service	014	Information applies to a capability which a service provider entity makes available to a service user entity through a set of interfaces that define a behaviour, such as a use case
collectionHardware	003	Information applies to the collection hardware class
nonGeographicDataset	007	Information applies to non-geographic data
dimensionGroup	008	Information applies to a dimension group

Req 52:*Each md_scope column value in a gpkg_metadata table or updateable view SHALL be one of the name column values from Table 11 in clause 2.4.2.1.2.*

2.4.3. Metadata Reference Table

2.4.3.1. Data

2.4.3.1.1. Table Definition

Req 53:*A GeoPackage file that contains a gpkg_metadata table SHALL contain a gpkg_metadata_reference table per clause 2.4.3.1.1, Table 12 and Table 34.*

The second component of GeoPackage metadata is the gpkg_metadata_reference table that links metadata in the gpkg_metadata table to data in the feature, and tiles tables defined in clauses 2.1.6 and2.2.7. The gpkg_metadata_reference table is not required to contain any rows.

Table 12 Metadata Reference Table or View Definition

Table or View Name: gpkg_metadata_reference					
Column Name	Col Type	Column Description	Null	Default	Key

[1] The “catalog” md_scope MAY be used for Feature Catalog [B40] information stored as XML metadata that is linked to features stored in a GeoPackage.

[2] The “schema” md_scope MAY be used for Application Schema [B37][B38][B39][B44] information stored as XML metadata that is linked to features stored in a GeoPackage.

[3] The “taxonomy” md_scope MAY be used for taxonomy or knowledge system [B41][B42] “linked data” information stored as XML metadata that is linked to features stored in a GeoPackage.

Field Cod

reference_scope	text	Metadata reference scope; one of ‘geopackage’, ‘table’,‘column’, ’row’, ’row/col’	no		
table_name	text	Name of the table to which this metadata reference applies, or NULL for reference_scope of ‘geopackage’.	yes		
column_name	text	Name of the column to which this metadata reference applies; NULL for reference_scope of ‘geopackage’,‘table’ or ‘row’, or the name of a column in the table_name table for reference_scope of ‘column’ or ‘row/col’	yes		
row_id_value1	integer	NULL for reference_scope of ‘geopackage’, ‘table’ or ‘column’, or the rowed of a row record in the table_name table for reference_scope of ‘row’ or ‘row/col’	yes		
timestamp	text	timestamp value in ISO 8601format as defined by the strftime function '%Y-%m-%dT%H:%M:%fZ' format string applied to the current time	no	strftime('%Y-%m-%dT%H:%M:%fZ', CURRENT_TIMESTAMP)	
md_file_id	integer	gpkg_metadata table id column value for the metadata to which this gpkg_metadata_reference applies	no		FK
md_parent_id	integer	gpkg_metadata table id column value for the hierarchical parent gpkg_metadata for the gpkg_metadata to which this gpkg_metadata_reference applies, or NULL if md_file_id forms the root of a metadata hierarchy	yes		FK

Every row in gpkg_metadata_reference that has null value as md_parent_id forms the root of a metadata hierarchy[2].

See Annex C: Table Definition SQL clause C.10 gpkg_metadata_reference.

2.4.3.1.2. Table Data Values

Req 54:*Every gpkg_metadata_reference table reference scope column value SHALL be one of ‘geopackage’, ‘table’,‘column’, ’row’, ’row/col’.*

Req 55:*Every gpkg_metadata_reference table row with a reference_scope column value of ‘geopackage’ SHALL have a table_name column value that is NULL. Every other*

[1] In SQLite, the rowid value is always equal to the value of a single-column primary key on an integer column [B30] and is not changed by a database reorganization performed by the VACUUM SQL command.

[2] Such a metadata hierarchy MAY have only one level of defined metadata.

gpkg_metadata_reference table row SHALL have a table_name column value that references a value in the gpkg_contents table_name column.

Req 56: ***Every gpkg_metadata_reference table row with a reference_scope column value of 'geopackage','table' or 'row' SHALL have a column_name column value that is NULL. Every other gpkg_metadata_reference table row SHALL have a column_name column value that contains the name of a column in the SQLite table or view identified by the table_name column value.***

Req 57: ***Every gpkg_metadata_reference table row with a reference_scope column value of 'geopackage', 'table' or 'column' SHALL have a row_id_value column value that is NULL. Every other gpkg_metadata_reference table row SHALL have a row_id_value column value that contains the ROWID of a row in the SQLite table or view identified by the table_name column value.***

Req 58: ***Every gpkg_metadata_reference table row timestamp column value SHALL be in ISO 8601 [41]format containing a complete date plus UTC hours, minutes, seconds and a decimal fraction of a second, with a 'Z' ('zulu') suffix indicating UTC.[1]***

Req 59: ***Every gpkg_metadata_reference table row md_file_id column value SHALL be an id column value from the gpkg_metadata table.***

Req 60: ***Every gpkg_metadata_reference table row md_parent_id column value that is NOT NULL SHALL be an id column value from the gpkg_metadata table that is not equal to the md_file_id column value for that row.***

2.5. Other Data Tables

Application-specific or vendor-specific data tables that do not conform to the table definitions or other requirements for storing feature or tile data or metadata as defined in other sections of this document MAY be defined and used to store supplemental information in a GeoPackage database. Such other data tables MAY be described by rows in the gpkg_spatial_ref_sys, gpkg_contents, gpkg_geometry_columns, gpkg_extensions, gpkg_data_columns, gpkg_tile_matrix_metadata, gpkg_metadata and gpkg_metadata_reference tables. GeoPackage applications MAY ignore such descriptive rows and other data tables they do not recognize

2.6. Extension Mechanism

The GeoPackage extension mechanism is an optional capability that is a required dependency for the optional registered extension capabilities defined in clause 3**.**

[1] The following statement selects an ISO 8601timestamp value using the SQLite strftime function:

```
    SELECT (strftime('%Y-%m-%dT%H:%M:%fZ','now')).
```

Field Cod

2.6.1. Extensions

2.6.1.1. Data

The optional capabilities defined in sub clauses and requirement statements of this clause MAY be implemented by any **GeoPackage file.**

2.6.1.1.1. Table Definition

Req 61:*A GeoPackage file MAY contain a table or updateable view named gpkg_extensions. If present this table SHALL be defined per clause2.6.1.1.1, Table 13 and Table 23.*

The gpkg_extensions table in a GeoPackage file registers extension capabilities required to make direct use of its contents. An application that access a GeoPackage file can query the gpkg_extensions table instead of the contents of an all of the user data tables to determine if it has the required capabilities, and to "fail fast" and return an error message if it does not.

Table 13 GeoPackage Extensions Table or View Definition

Table or View Name: gpkg_extensions				
Column Name	**Col Type**	**Column Description**	**Null**	**Key**
table_name	text	Name of the table that requires the extension. When NULL, the extension is required for the entire GeoPackage. SHALL NOT be NULL when the column_name is not NULL.	**yes**	**Unique**
column_name	text	Name of the column that requires the extension. When NULL, the extension is required for the entire table.	**yes**	**Unique**
extension_name	text	The name of the extension that is required, in the form <author>_<extension name>.	**no**	**Unique**

See Annex C: Table Definition SQL clause C.3 gpkg_extensions.

2.6.1.1.2. Table Data Values

Req 62:*Values of the gpkg_extensions table table_name column SHALL reference values in the gpkg_contents table_name column or be NULL. They SHALL NOT be NULL for rows where the column_name value is not NULL.*

Req 63:*The column_name column value in a gpkg_extensions table row SHALL be the name of a column in the table specified by the table_name column value for that row, or be NULL.*

Field Cod

Req 64:*Each extension_name column value in a gpkg_extensions table row SHALL be a unique value of the form <author>_<extension name> where <author> indicates the person or organization that developed and maintains the extension. The valid character set for <author> SHALL be [a-zA-Z0-9]. The valid character set for <extension name> SHALL be [a-zA-Z0-9_]. An extension_name for the "gpkg" author name SHALL be one of those listed in Table 14.*

The author value "gpkg" is reserved for GeoPackage extensions that are developed and maintained by OGC. Requirements for extension names for the "gpkg" author name are defined in the clauses listed in Table 14 below. GeoPackage implementers use their own author names to register other extensions.[1]

Table 14 Reserved GeoPackage Extensions

Extension Name	Defined in Clause	Description
gpkg_geom_<gname>	3.1.2.1.2	<gname> is name of specific extension geometry type from Annex G Table 47
gpkg_zoom_other	3.2.1.1.1	Tile matrix set user data table with zoom intervals other than powers of two between adjacent layers
gpkg_webp	3.2.2.2.1	Raster images in image/webp format
gpkg_tiff	3.2.3.1.2	Raster images in image/tiff format
gpkg_nitf	3.2.4.1.2	Raster images in application/vnd.NITF
gpkg_rtree_index	3.1.3.1.2	Rtree spatial index implemented via SQLite Virtual Table and maintained via triggers that use all minimal runtime SQL functions except ST_SRID.
gpkg_gtype_trigger	3.1.4.1.2	Trigger to restrict GeoPackage feature table geometry column geometries to geometry types equal to or substitutable for the geometry type specified for the column in the gpkg_geometry_columns table geometry_type_name column.
gpkg_srs_id_trigger	3.1.5.1.2	Trigger to restrict GeoPackage feature table geometry column geometries to the srs_id specified for the geometry column in the gpkg_geometry_columns table using the ST_SRID SQL function.

[1] See Requirement 67 in Clause 3.

Field Cod

2.6.1.2. API

The optional capabilities defined in sub clauses and requirement statements of this clause MAY be implemented by any **GeoPackage SQLite Extension.**

2.6.1.2.1. API GeoPackage SQLite Config

Req 65:*A GeoPackage SQLite Extension MAY provide SQL runtime functions or rtree support for a GeoPackage file. One that does so SHALL have the SQLite library compile and run time options specified in clause 2.6.1.2.1 Table 15*

Table 15 API GeoPackage SQLite Configuration

Setting	Option	Shall / Not (Value)	Discussion
compile	SQLITE_OMIT_LOAD_EXTENSION	Not	The load_extension() function is required to implement the MinimalRuntimeSQLFunctions
compile	SQLITE_OMIT_VIRTUALTABLE	Not	Virtual tables are required to implement RTrees
compile	SQLITE_ENABLE_RTREE	Shall	Rtrees are used for GeoPackage Spatial Indexes. See SpatialIndexRequirements
compile	SQLITE_RTREE_INT_ONLY	Not	Rtrees with floating point values are used for GeoPackage Spatial Indexes.

2.6.1.2.2. Safe GeoPackage SQLite Config

Req 66:*A GeoPackage SQLite Extension MAY provide primary/foreign key and trigger support for a GeoPackage file. One that does so SHALL have the SQLite library compile and run time options specified in clause 2.6.1.2.2 Table 16.*

Table 16 Safe GeoPackage SQLite Configuration

Setting	Option	Shall / Not (Value)	Discussion
compile	SQLITE_DEFAULT_FOREIGN_KEYS	Shall (1)	Foreign key constraints are used to maintain GeoPackage relational integrity.

Field Cod

compile	SQLITE_OMIT_FOREIGN_KEY	Not	Foreign key constraints are used to maintain GeoPackage relational integrity.
run	PRAGMA foreign_keys	Not (OFF)	Foreign key constraints are used to maintain GeoPackage relational integrity.
compile	SQLITE_OMIT_INTEGRITY_CHECK	Not	This option omits support for the integrity_check pragma, which does an integrity check of the entire database. This pragma should be part of GeoPackage conformance validation.
compile	SQLITE_OMIT_SUBQUERY	Not	This option omits support for sub-selects and the IN() operator, both of which are used in GeoPackage? triggers.
compile	SQLITE_OMIT_TRIGGER	Not	Defining this option omits support for TRIGGER objects. Neither the CREATE TRIGGER or DROP TRIGGER commands are available in this case, and attempting to execute either will result in a parse error. This option also disables enforcement of foreign key constraints, since the code that implements triggers and which is omitted by this option is also used to implement foreign key actions. Foreign keys and triggers are used by Safe GeoPackages. Triggers are used to maintain spatial indexes.

Registered Extensions

The optional registered extension capabilities specified in this clause depend on required capabilities specified in clause 1 and optional capabilities specified in clause 2.. Any optional

Field Cod

capability in this class that is implemented SHALL be registered using the Extension Mechanism defined in clause 2.6, as specified by the sub clauses below for extension name and extension row for the optional capability.

Req 67:***Any table in a GeoPackage file subject to a registered extension with an author_name other than "gpkg" SHALL NOT be described by a gpkg_contents table row with a data_type value of "feature" or "tiles".***

Such tables do not count[1] in the determination of GeoPackage file validity per clause 2 Req 13: but otherwise MAY differ from "features" and "tiles" tables only with respect to non-gpkg registered extensions that enable implementers to meet requirements not addressed in the current version of this specification, and may be included in a valid GeoPackage.

3.1. Features

3.1.1. Geometry Encoding

3.1.1.1. Data

The optional registered extension capabilities defined in sub clauses and requirement statements of this clause MAY be implemented by any **GeoPackage file** that implements the requirements of clause 2.1 and clause 2.6.1.1 on which they depend**.**

3.1.1.1.1. BLOB Format

A GeoPackage file CAN store geometries using BLOB formats other than the GeoPackageBinary format specified in clause 2.1.3.1.1. However, tables with geometry column data encoded in such extension formats are not considered to be GeoPackage feature tables for purposes of determining GeoPackage file validity.

3.1.1.1.2. BLOB format - Extensions Name

Req 68:***An extension name in the form <author_name>_geometry encoding SHALL be defined for an author name other than "gpkg" for each geometry BLOB format other than GeoPackageBinary used in a GeoPackage file.***

3.1.1.1.3. BLOB format - Extensions Row

Req 69:***A GeoPackage file that contains geometry column BLOB values encoded in an extension format SHALL contain a gpkg_extensions table that contains row records with table_name and column_name values from the gpkg_geometry_columns table row records that identify extension format uses, and with extension_name column values constructed per clause 3.1.1.1.2.***

[1] They MAY be contained in a valid GeoPackage file.

Field Cod

3.1.2. Geometry Types

3.1.2.1. Data

The optional extension capabilities defined in sub clauses and requirement statements of this clause MAY be implemented by any **GeoPackage file** that implements the requirements of clause 2.1 and clause 2.6.1.1on which they depend, as specified in Annex A**.**

3.1.2.1.1. Extension Types

Req 70:*A GeoPackage file MAY store feature table geometries with the extended non-linear geometry types (CircularString, CompoundCurve, CurvePolygon, MultiCurve, MultiSurface, Curve, Surface) in Annex G. If it does so, they SHALL be encoded in the GeoPackageBinary geometry format.*

3.1.2.1.2. Extensions Types - Extensions Name

Req 71:*An extension name to specify a feature geometry extension type SHALL be defined for the "gpkg" author name using the "gpkg_geom_<gname>" template where <gname> is the name of the extension geometry type from Annex G used in a GeoPackage file.*

3.1.2.1.3. Extensions Types - Extensions Row

Req 72:*A GeoPackage file that contains a gpkg_geometry_columns table or updateable view with row records that specify extension geometry_type column values SHALL contain a gpkg_extensions table that contains row records with table_name and column_name values from the geometry_columns row records that identify extension type uses, and extension_name column values for each of those geometry types constructed per clause 3.1.2.1.2.*

3.1.3. Spatial Indexes

3.1.3.1. Data

The optional registered extension capabilities defined in sub clauses and requirement statements of this clause MAY be implemented by any **GeoPackage file** that implements requirements of clause 2.1 and clause 2.6 on which they depend**.**

3.1.3.1.1. Spatial Indexes Implementation

Spatial indexes provide a significant performance advantage for searches with basic envelope spatial criteria that return subsets of the rows in a feature table with a non-trivial number (thousands or more) of rows.[1]

Spatial indexes are an optional GeoPackage extension with the following runtime requirements

- The SQLite RTree extension SHALL be available and loaded per clause 2.6.1.2.1.

[1]If an application process will make many updates, it is often faster to drop the indexes, do the updates, and then recreate the indexes.

Field Cod

- SQLite trigger support SHALL be present per clause 2.6.1.2.2.
- The ST_MinX, ST_MaxX, ST_MinY, ST_MaxY SQL functions specified in Annex D SHALL be available.

Req 73:*A GeoPackage file SHALL implement spatial indexes on feature table geometry columns as specified in clause 3.1.3.1.1 using the SQLite Virtual Table RTrees and triggers specified in Annex E.*

3.1.3.1.2. Spatial Indexes - Extensions Name

Req 74:*The "gpkg_rtree_index" extension name SHALL be used as a gpkg_extension table extension name column value to specify implementation of spatial indexes on a geometry column.*

3.1.3.1.3. Spatial Indexes - Extensions Row

Req 75:*A GeoPackage file that implements spatial indexes SHALL have a gpkg_extensions table that contains a row for each spatially indexed column with extension_name "gpkg_rtree_index", the table_name of the table with a spatially indexed column, and the column_name of the spatially indexed column.*

3.1.4. Geometry Type Triggers

3.1.4.1. Data

The optional extension capabilities defined in sub clauses and requirement statements of this clause MAY be implemented by any **GeoPackage file** that implements requirements of clause 2.1 and clause 2.6 on which they depend**.**

3.1.4.1.1. Geometry Type Triggers – Implementation

Req 76:*A GeoPackage file SHALL include the SQL insert and update triggers specified in clause 3.1.4.1.1 and Table 17 on every feature table geometry column to enforce the geometry type values specified for those columns in the gpkg_geometry_columns table.*

Table 17 Geometry Type Trigger Definition SQL Template

```
CREATE TRIGGER fgti_<t>_<c> BEFORE INSERT ON '<t>'
FOR EACH ROW
BEGIN
  SELECT RAISE (ROLLBACK, 'insert on <t> violates constraint:
ST_GeometryType(<c>) is not assignable from
gpkg_geometry_columns.geometry_type_name value')
  WHERE (SELECT geometry_type_name FROM gpkg_geometry_columns
         WHERE Lower(table_name) = Lower('<t>')
         AND   Lower(column_name) = Lower('<c>')
         AND   gpkg_IsAssignable(geometry_type_name,
                                 ST_GeometryType(NEW.<c>)) = 0;

END
```

Field Cod

```
CREATE TRIGGER fgtu_<t>_<c> BEFORE UPDATE OF '<c>' ON '<t>'
FOR EACH ROW
BEGIN
SELECT RAISE (ROLLBACK,
'update of <c> on <t> violates constraint: ST_GeometryType(<c>) is
not assignable from gpkg_geometry_columns.geometry_type_name value')
WHERE (SELECT geometry_type_name FROM gpkg_geometry_columns
       WHERE Lower(table_name) = Lower('<t>')
       AND   Lower(column_name) = Lower('<c>')
       AND   gpkg_IsAssignable(geometry_type_name,
                               ST_GeometryType(NEW.<c>)) = 0;
END
```

where <t> and <c> are replaced with the names of the feature table and geometry column being inserted or updated.

The triggers to enforce gpkg_geometry_columns geometry_type_name constraints on feature table geometries use Minimal Runtime SQL Functions specified in Annex D.

3.1.4.1.2. Geometry Type Triggers – Extensions Name

Req 77:*The "gpkg_geometry_type_trigger" extension name SHALL be used as a geopackage_extension table extension name column value to specify implementation of geometry type triggers.*

3.1.4.1.3. Geometry Type Triggers – Extensions Row

Req 78:*A GeoPackage file that implements geometry type triggers on feature table geometry columns SHALL contain a gpkg_extensions table that contains a row for each such geometry column with extension_name "gpkg_geometry_type_trigger", table_name of the feature table with a geometry column, and column_name of the geometry column.*

3.1.5. SRS_ID Triggers

3.1.5.1. Data

The optional extension capabilities defined in sub clauses and requirement statements of this clause MAY be implemented by any **GeoPackage file** that implements requirements of clause 2.1 and clause 2.6 on which they depend**.**

3.1.5.1.1. SRS_ID Triggers – Implementation

Req 79:*A GeoPackage file SHALL include the SQL insert and update triggers specified in clause 3.1.5.1.1 and Table 18 on every feature table geometry column to enforce the srs_id values specified for those columns in the gpkg_geometry_columns table.*

Table 18 SRS_ID Trigger Definition SQL Templates

```
CREATE TRIGGER fgsi_<t> _<c> BEFORE INSERT ON '<t>'
FOR EACH ROW
BEGIN
```

Field Cod

```
  SELECT RAISE (ROLLBACK, 'insert on <t>violates constraint:
ST_SRID(<c>) does not match gpkg_geometry_columns.srs_id value')
  WHERE (SELECT srs_id FROM gpkg_geometry_columns
       WHERE Lower(table_name) = Lower('<t>')
      AND   Lower(column_name) = Lower('<c>')
      AND   ST_SRID(NEW.'<c>') <> srs_id) ;
END

CREATE TRIGGER fgsu_<t>_<c> BEFORE UPDATE OF '<c>' ON '<t>'
FOR EACH ROW
BEGIN
SELECT RAISE (ROLLBACK,
'update of <c> on <t> violates constraint: ST_SRID(<c>) does not
match gpkg_geometry_columns.srs_id value')
WHERE (SELECT srs_id FROM gpkg_geometry_columns
       WHERE Lower(table_name) = Lower('<t>')
      AND   Lower(column_name) = Lower('<c>')
      AND   ST_SRID(NEW.'<c>') <> srs_id);
END
```

where <t> and <c> are replaced with the names of the feature table and geometry column being inserted or updated.

The triggers to enforce geometry_columns srs_id constraints on feature table geometries use Minimal Runtime SQL Functions specified in Annex F

3.1.5.1.2. SRS_ID Triggers – Extensions Name

Req 80:***The "gpkg_srs_id_trigger" extension name SHALL be used as a geopackage_extension table extension name column value to specify implementation of SRS_ID triggers.***

3.1.5.1.3. SRS_ID Triggers – Extensions Row

Req 81:***A GeoPackage file that implements srs_id triggers on feature table geometry columns SHALL contain a gpkg_extensions table that contains a row for each geometry column with extension_name "gpkg_srs_id_trigger", table_name of the feature table with a geometry column, and column_name of the geometry column.***

3.2. Tiles

3.2.1. Zoom Levels

3.2.1.1. Data

The optional extension capabilities defined in sub clauses and requirement statements of this clause MAY be implemented by any **GeoPackage file** that implements the requirements of clause 2.2 and clause 2.6.1.1 on which they depend**.**

Field Cod

3.2.1.1.1. Zoom Other Intervals

As a registered extension, a GeoPackage file MAY contain tile matrix set user data tables with pixel sizes that vary by irregular intervals or by regular intervals other than powers of two (the default) between adjacent zoom levels, as described in the gpkg_tile_matrix_metadata table.

3.2.1.1.2. Zoom Other – Extensions Name

Req 82: ***The "gpkg_zoom_other" extension name SHALL be used as a gpkg_extension table extension name column value to specify implementation of other zoom intervals on a tile matrix set user data table.***

3.2.1.1.3. Zoom Other – Extensions Row

Req 83: ***A GeoPackage file that implements other zoom intervals SHALL have a gpkg_extensions table that contains a row for each tile matrix set user data table with other zoom intervals with extension_name "gpkg_zoom_other", the table_name of the table with other zoom intervals, and the "tile_data" column_name.***

3.2.2. Tile Encoding WEBP

3.2.2.1. Data

The optional extension capabilities defined in sub clauses and requirement statements of this clause MAY be implemented by any **GeoPackage file** that implements the requirements of clause 2.2 and clause 2.6.1.1 on which they depend**.**

3.2.2.2. WEBP MIME Type

As a registered extension, a GeoPackage file that contains a tile matrix user data table that contains tile data MAY store tile_data in MIME type image/x-webp[26].

3.2.2.2.1. WEBP -- Extensions Name

Req 84: ***The "gpkg_webp" extension name SHALL be used as a geopackage_extension table extension name column value to specify storage of raster images in WEBP format.***

3.2.2.2.2. WEBP -- Extensions Row

Req 85: ***A GeoPackage file that contains tile matrix user data tables with tile_data columns that contain raster images in WEBP format SHALL contain a gpkg_extensions table that contains row records with table_name values for each such table, "tile_data" column_name values and extension_name column values of "gpkg_webp".***

Field Cod

3.2.3. Tiles Encoding TIFF

3.2.3.1. Data

The optional extension capabilities defined in sub clauses and requirement statements of this clause MAY be implemented by any **GeoPackage file** that implements the requirements of clause 2.2 and clause 2.6.1.1 on which they depend**.**

3.2.3.1.1. TIFF MIME Type

As a registered extension, a GeoPackage file that contains a tile matrix user data table that contains tile data MAY store tile_data in MIME type image/tiff [27] for GeoTIFF images [28][29] that meet the requirements of the NGA Implementation Profile [31] for coordinate transformation case 3 where the position and scale of the data is known exactly, and no rotation of the image is required.

3.2.3.1.2. TIFF -- Extensions Name

Req 86:*The "gpkg_tiff" extension name SHALL be used as a geopackage_extension table extension name column value to specify storage of raster images in TIFF format.*

3.2.3.1.3. Extensions Row

Req 87:*A GeoPackage file that contains tile matrix user data tables with tile_data columns that contain raster images in TIFF format per SHALL contain a gpkg_extensions table that contains row records with table_name values for each such table, "tile_data" column_name values and extension_name column values of "gpkg_tiff".*

3.2.4. Tile Encoding NITF

3.2.4.1. Data

The optional extension capabilities defined in sub clauses and requirement statements of this clause MAY be implemented by any **GeoPackage file** that implements the requirements of clause 2.2and clause 2.6.1.1 on which they depend**.**

3.2.4.1.1. NITF MIME Type

As a registered extension, a GeoPackage file that contains a tile matrix user data table that contains tile data MAY store tile_data in MIME type application/vnd.NITF[46] for National Imagery Transmission Format images.

3.2.4.1.2. NITF -- Extensions Name

Req 88:*The "gpkg_nitf" extension name SHALL be used as a geopackage_extension table extension name column value to specify storage of raster images in NITF format.*

Field Cod

3.2.4.1.3. NITF -- Extensions Row

Req 89:*A GeoPackage file that contains tile matrix user data tables with tile_data columns that contain raster images in NITF format SHALL contain a gpkg_extensions table that contains row records with table_name values for each such table, "tile_data" column_name values and extension_name column values of "gpkg_nitf".*

3.2.5. Tile Encoding Other

3.2.5.1. Data

The optional extension capabilities defined in sub clauses and requirement statements of this clause MAY be implemented by any **GeoPackage file** that implements the requirements of clause 2.2 and clause 2.6.1.1 on which they depend**.**

3.2.5.1.1. Other MIME Type

As a registered extension, a GeoPackage file that contains a tile matrix user data table that contains tile data MAY store tile_data in other MIME types. However, a table with such data is not considered to be a "tiles" table for purposes of determining GeoPackage file validity.

3.2.5.1.2. Other Extensions Name

Req 90:*An extension name in the form <author_name>_<other>_mime_type SHALL be defined for an author name other than "gpkg" for each other MIME image format used for tile_data columns in tile matrix set user data tables, where <other> is replaced by the other MIME type abbreviation in uppercase*

3.2.5.1.3. Other Extensions Row

Req 91:*A GeoPackage file that contains tile matrix user data tables with tile_data columns that contain raster images in a MIME type format other than those defined in this specification SHALL contain a gpkg_extensions table that contains row records with table_name values for each such table, "tile_data" column_name values and extension_name column values of the other format extension name defined per clause 3.2.5.1.2.*

3.3. Any Tables

3.3.1. Other Trigger

3.3.1.1. Data

The optional extension capabilities defined in sub clauses and requirement statements of this clause MAY be implemented by any **GeoPackage file** that implements requirements of clause 2.1 and/or 2.2 and clause 2.6 on which they depend.

3.3.1.1.1. Other Trigger Implementation

As a registered extension, GeoPackage files MAY contain other triggers that require support from GeoPackage SQLite Extension functions other than those provided by SQLite or the

Field Cod

GeoPackage Minimal Runtime SQL Functions to enforce data integrity or application business rule constraints. [1]

3.3.1.1.2. Other Trigger – Extensions Name

Req 92:*An extension name in the form <author><extension> for an author name other than "gpkg" SHALL be defined as a geopackage_extension table extension name column value to specify triggers in a GeoPackage file that use SQL functions other than those provided by SQLite or the GeoPackage Minimal Runtime SQL Functions.*

3.3.1.1.3. Other Trigger – Extensions Row

Req 93:*A GeoPackage file that implements triggers that use SQL functions other than those provided by SQLite or the GeoPackage Minimal Runtime SQL Functions SHALL have a gpkg_extensions table that contains row records with table_name values for each such table, column_name values for each such column and extension_name column values of the other trigger extension name defined per clause 3.3.1.1.2.*

Security Considerations

Security considerations for implementations utilizing GeoPackages are in the domain of the implementing application, deployment platform, operating system and networking environment. The GeoPackage specification does not place any constraints on application, platform, operating system level or network security

[1] Other triggers that rely only on SQLite and Minimal Runtime SQL functions do not need to be registered as an extension.

Field Cod

Annex A Conformance / Abstract Test Suite (Normative)

A.1 Base

A.1.1 Core

A.1.1.1 SQLite Container

A.1.1.1.1 Data

A.1.1.1.1.1 File Format

Test Case ID: /base/core/container/data/file_format
Test Purpose: Verify that the Geopackage file is an SQLite version_3 database
Test Method: Pass if the first 16 bytes of the file contain “SQLite format 3” in ASCII.
Reference: Clause 1.1.1.1.1 Req 1:
Test Type: Basic

A.1.1.1.1.2 File Extension Name

Test Case ID: /base/core/container/data/file_extension_name
Test Purpose: Verify that the geopackage file extension is ".gpkg"
Test Method: Pass if the geopackage file extension is ".gpkg"
Reference: Clause 1.1.1.1.2 Req 2:
Test Type: Basic

A.1.1.1.1.3 Integrity Check

Test Case ID: /base/core/container/data/file_integrity
Test Purpose: Verify that the geopackage file passes the SQLite integrity check.
Test Method: Pass if PRAGMA integrity_check returns “ok"
Reference: Clause 1.1.1.1.3 Req 3:
Test Type: Capability

A.1.1.1.2 API

A.1.1.1.2.1 Structured Query Language

Test Case ID: /base/core/container/api/sql
Test Purpose: Test that the GeoPackage SQLite Extension provides the SQLite SQL API interface.
Test Method:

1) SELECT * FROM sqlite_master
2) Fail if returns an SQL error.
3) Pass otherwise

Reference: Clause 1.1.1.2.1 Req 4:
Test Type: Capability

A.1.1.1.2.2 Every GPKG SQLite Configuration

Test Case ID: /base/core/container/api/every_gpkg_sqlite_config
Test Purpose: Verify that a GeoPackage SQLite Extension has the Every GeoPackage SQLite Configuration compile and run time options.
Test Method:

1) SELECT sqlite_compileoption_used**(**'SQLITE_OMIT_AUTOINCREMENT'**)**

Field Cod

2) Fail if returns 1
3) SELECT sqlite_compileoption_used('SQLITE_OMIT_DATETIME_FUNCS')
4) Fail if returns 1
5) SELECT sqlite_compileoption_used('SQLITE_OMIT_ DEPRECATED ')
6) Fail if returns 0
7) SELECT sqlite_compileoption_used('SQLITE_OMIT_ FLOATING_POINT ')
8) Fail if returns 1
9) SELECT sqlite_compileoption_used('SQLITE_OMIT_PRAGMA')
10) Fail if returns 1
11) SELECT sqlite_compileoption_used('SQLITE_OMIT_FLAG_PRAGMAS')
12) Fail if returns 1
13) SELECT sqlite_compileoption_used('SQLITE_OMIT_VIEW')
14) Fail if returns 1
15) Pass otherwise

Reference: Clause 1.1.1.2.2 Req 5:
Test Type: Basic

A.1.1.2 Spatial Reference Systems

A.1.1.2.1 Data

A.1.1.2.1.1 Table Definition

Test Case ID: /base/core/spatial_ref_sys/data/table_def
Test Purpose: Verify that the spatial_ref_sys table exists and has the correct definition.
Test Method:

1) SELECT sql FROM sqlite_master WHERE type = 'table' AND tbl_name = 'gpkg_spatial_ref_sys'
2) Fail if returns an empty result set
3) Pass if column names and column definitions in the returned CREATE TABLE statement in the sql column value, including data type, nullability, default values and primary, foreign and unique key constraints match all of those in the contents of C.1 Table 19. Column order, check constraint and trigger definitions, and other column definitions in the returned sql are irrelevant.
4) Fail otherwise.

Reference: Clause 1.1.2.1.1 Req 6:
Test Type: Basic

A.1.1.2.1.2 Table Data Values

Test Case ID: /base/core/spatial_ref_sys/data_values_default
Test Purpose: Verify that the spatial_ref_sys table contains the required default contents.
Test Method:

1) SELECT srid, auth_name, auth_srid, srtext FROM spatial_ref_sys WHERE srid = -1 returns -1 “NONE” -1 “Undefined”, AND
2) SELECT srid, auth_name, auth_srid, srtext FROM spatial_ref_sys WHERE srid = 0 returns 0 “NONE” 0 “Undefined”, AND
3) SELECT srid, auth_name, auth_srid, srtext FROM spatial_ref_sys WHERE srid = 4326 returns 4326 epsg 4326 GEOGCS["WGS 84", DATUM["WGS_1984",

Field Cod

SPHEROID["WGS 84",6378137,298.257223563, AUTHORITY["EPSG","7030"]], AUTHORITY["EPSG","6326"]],PRIMEM["Greenwich",0,AUTHORITY["EPSG","8901"]],UNIT["degree",0.0174532925199433,AUTHORITY["EPSG","9122"]], AUTHORITY["EPSG","4326"]] (whitespace irrelevant)

4) Pass if tests 1-3 are met
5) Fail otherwise

Reference: Clause 1.1.2.1.2 Req 7:
Test Type: Capability

Test Case ID: /base/core/spatial_ref_sys/data_values_required
Test Purpose: Verify that the spatial_ref_sys table contains rows to define all srs_id values used by features and tiles in a GeoPackage.
Test Method:

1) SELECT DISTINCT gc.srs_id AS gc_srid, srs.srs_name, srs.srs_id, srs.organization, srs.organization_coordsys_id, srs.definition FROM gpkg_contents AS gc LEFT OUTER JOIN gpkg_spatial_ref_sys AS srs ON srs.srs_id = gc.srs_id
2) Pass if no returned srs values are NULL.
3) Fail otherwise

Reference: Clause Clause 1.1.2.1.2 Req 7:

A.1.1.3 Test Type: CapabilityContents

A.1.1.3.1 Data

A.1.1.3.1.1 Table Definition

Test Case ID: /base/core/contents/data/table_def
Test Purpose: Verify that the gpkg_contents table exists and has the correct definition.
Test Method:

1) SELECT sql FROM sqlite_master WHERE type = 'table' AND tbl_name = 'gpkg_contents'
2) Fail if returns an empty result set.
3) Pass if the column names and column definitions in the returned CREATE TABLE statement, including data type, nullability, default values and primary, foreign and unique key constraints match all of those in the contents of C.2 Table 20. Column order, check constraint and trigger definitions, and other column definitions in the returned sql are irrelevant.
4) Fail Otherwise

Reference: Clause 1.1.3.1.1 Req 9:
Test Type: Basic

A.1.1.3.1.2 Table Data Values

Test Case ID: /base/core/contents/data/data_values_table_name
Test Purpose: Verify that the table_name column values in the gpkg_contents table are valid.
Test Method:

Field Cod

1) SELECT DISTINCT gc.table_name AS gc_table, sm.tbl_name
   FROM gpkg_contents AS ge LEFT OUTER JOIN sqlite_master AS sm ON gc.table_name = sm.tbl_name
2) Not testable if returns an empty result set.
3) Fail if any gpkg_contents.table_name value is NULL
4) Pass otherwise.

Reference: Clause 1.1.3.1.2 Req 10:
Test Type: Capability

Test Case ID: /base/core/contents/data/data_values_last_change
Test Purpose: Verify that the gpkg_contents table last_change column values are in ISO 8601 [41]format containing a complete date plus UTC hours, minutes, seconds and a decimal fraction of a second, with a 'Z' ('zulu') suffix indicating UTC.
Test Method:

1) SELECT last_change from gpkg_contents.
2) Not testable if returns an empty result set.
3) For each row from step 1
   a. Fail if format of returned value does not match yyyy-mm-ddThh:mm:ss.hhhZ
   b. Log pass otherwise
4) Pass if logged pass and no fails.

Reference: Clause 1.1.3.1.2 Req 11:
Test Type: Capability

Test Case ID: /base/core/contents/data/data_values_srs_id
Test Purpose: Verify that the gpkg_contents table srs_id column values are defined in the gpkg_spatial_ref_sys table.
Test Method:

1) SELECT srs_id FROM gpkg_contents
2) Not testable if returns an empty result set
3) SELECT srs_id FROM gpkg_contents WHERE srs_id NOT IN (SELECT srs_id FROM gpkg_spatial_ref_sys)
4) Fail if does not return an empty result set
5) SELECT srs_id FROM gpkg_contents gc WHERE data_type = 'features' AND srs_id NOT IN (SELECT srs_id FROM gpkg_geometry_columns WHERE table_name = gc.table_name)
6) Fail if does not return an empty result set
7) Pass otherwise

Reference: Clause Clause 1.1.3.1.2 Req 12:
Test Type: Capability

A.2 Options

Test Case ID: /opt/valid_geopackage

Field Cod

Test Purpose: Verify that a GeoPackage contains a features or tiles table and gpkg_contents table row describing it.
Test Method:

1) Execute test /opt/features/contents/data/features_row
2) Pass if test passed
3) Execute test /opt/tiles/contents/data/tiles_row
4) Pass if test passed
5) Fail otherwise

Reference: Clause 2 Req 13:
Test Type: Capability

A.2.1 Features

A.2.1.1 Contents

A.2.1.1.1 Data

A.2.1.1.1.1 Contents Table Feature Row

Test Case ID: /opt/features/contents/data/features_row
Test Purpose: Verify that the gpkg_contents table_name value table exists, and is apparently not a tiles table for every row with a data_type column value of "features"
Test Method:

1) SELECT table_name FROM gpkg_contents where data_type="features"
2) Fail if returns empty result set
3) For each row from step 1
    a. PRAGMA table_info(table_name)
    b. Fail if returns an empty result set
    c. Fail if result set contains four rows where the name column values are "zoom_level","tile_column","tile_row", and "tile_data"
    d. Fail if result set does not contain one row where the pk column value is 1 and the type column value is "INTEGER"
4) Pass if no fails

Reference: Clause 2.1.2.1.1 Req 14:
Test Type: Capability

A.2.1.2 Geometry Encoding

A.2.1.2.1 Data

A.2.1.2.1.1 BLOB Format

Test Case ID: /opt/features/geometry_encoding/data/blob
Test Purpose: Verify that geometries stored in feature table geometry columns are encoded in the GeoPackageBinary format.
Test Method:

1) SELECT table_name AS tn, column_name AS cn FROM gpkg_geometry_columns WHERE table_name IN (SELECT table_name FROM gpkg_contents WHERE data_type = 'features')
2) Not testable if returns an empty result set
3) For each row from step 1

Field Cod

   a. SELECT cn FROM tn
   b. Not testable if none found
   c. For each cn value from step a
      i. Fail if the first three bytes of each gc are not "GPB"
      ii. Fail if gc.version_number is not 0
      iii. Fail if ST_IsEmpty(cn value) = 1 and gc.flags.envelope != 0 and envelope values are not NaN
4) Pass if no fails

Reference: Clause 2.1.3.1.1 Req 15:
Test Type: Capability

A.2.1.2.2 API

A.2.1.2.2.1 Minimal Runtime SQL Functions

Test Case ID: /opt/features/geometry_encoding/sql_func
Test Purpose: Verify that a GeoPackage SQLite Extension provides the GeoPackage Minimal Runtime SQL functions.
Test Method:

1) Open Geometry Test Data Set GeoPackage with GeoPackage SQLite Extension
2) For each Geometry Test Data Set <gtype_test> data table row for each assignable (gtype, atype) and non-assignable (ntype, atype) combination of geometry type in Annex G, for an assortment of srs_ids, for an assortment of coordinate values, without and with z and / or m values, in both big and little endian encodings:
   a. SELECT 'Fail' FROM <gtype_test> WHERE ST_SRID(geom) != srs_id
   b. SELECT 'Fail' FROM <gtype_test> WHERE ST_GeometryType(geom) != atype
   c. SELECT 'Fail' FROM <gtype_test> WHERE GPKG_IsAssignable(gtype, atype) = 0
   d. SELECT 'Fail' FROM <gtype_test> WHERE GPKG_IsAssignable(ntype, atype) = 1
   e. SELECT 'Fail' FROM <gtype_test> WHERE ST_Is3D(geom) != is3d
   f. SELECT 'Fail' FROM <gtype_test> WHERE ST_IsMeasured(geom) != ism
   g. SELECT 'Fail' FROM <gtype_test> WHERE ST_MinX(geom) != minx
   h. SELECT 'Fail' FROM <gtype_test> WHERE ST_MaxX(geom) != maxx
   i. SELECT 'Fail' FROM <gtype_test> WHERE ST_MinY(geom) != miny
   j. SELECT 'Fail' FROM <gtype_test> WHERE ST_MaxY(geom) != maxy
   k. SELECT 'Fail' FROM <gtype_test> WHERE ST_MinZ(geom) != minz
   l. SELECT 'Fail' FROM <gtype_test> WHERE ST_MaxZ(geom) != maxz
   m. SELECT 'Fail' FROM <gtype_test> WHERE ST_MinM(geom) != minm
   n. SELECT 'Fail' FROM <gtype_test> WHERE ST_MaxM(geom) != maxm
3) Pass if no 'Fail' selected from step 2

Reference: Clause 2.1.3.2.1 Req 16:
Test Type: Capability

A.2.1.3 Geometry Types

A.2.1.3.1 Data

A.2.1.3.1.1 Core Types

Field Cod

Test Case ID: /opt/features/geometry_encoding/data/core_types_existing_sparse_data
Test Purpose: Verify that existing basic simple feature geometries are stored in valid GeoPackageBinary format encodings.
Test Method:

1) SELECT table_name FROM gpkg_geometry_columns
2) Not testable if returns an empty result set
3) SELECT table_name AS tn, column_name AS cn FROM gpkg_geometry_columns WHERE table_name IN (SELECT table_name FROM gpkg_contents WHERE data_type = 'features'),
4) Fail if returns an empty result set
5) For each row from step 3
    a. SELECT cn FROM tn;

    b. For each row from step a, if bytes 2-5 of cn.wkb as uint32 in endianness of gc.wkb byte 1of cn from #1 are a geometry type value from Annex G Table 46, then
        i. Log cn.header values, wkb endianness and geometry type
        ii. If cn.wkb is not correctly encoded per ISO 13249-3 clause 5.1.46 then log fail
        iii. If cn.flags.E is 1 - 4 and some cn.wkbx is outside of cn.envelope.minx,maxx then log fail
        iv. If cn.flags.E is 1 - 4 and some gc.wkby is outside of cn.envelope.miny,maxy then log fail
        v. If cn.flags.E is 2,4 and some gc.wkb.z is outside of cnenvelope.minz,maxz then log fail
        vi. If cn.flags.E is 3,4 and some gc.wkb.m is outside of cn.envelope.minm,maxm then log fail
        vii. If cn.flags.E is 5-7 then log fail
        viii. Otherwise log pass
6) Pass if log contanins pass and no fails

Reference: Clause 2.1.4.1.1 Req 17:
Test Type: Capability

Test Case ID: /opt/features/geometry_encoding/data/core_types_all_types_test_data
Test Purpose: Verify that all basic simple feature geometry types and options are stored in valid GeoPackageBinary format encodings.
Test Method:

1) Open GeoPackage that has feature geometry values of geometry type in Annex G, for an assortment of srs_ids, for an assortment of coordinate values, without and with z and / or m values, in both big and little endian encodings:
2) /opt/features/geometry_encoding/data/core_types_existing_sparse_data
3) Pass if log contains pass record for big and little endian GPB headers containing big and little endian WKBs for 0-1 envelope contents indicator codes for every geometry type value from Annex G Table 46 without and with z and/or m values.

Field Cod

4) Fail otherwise

Reference: Clause 2.1.4.1.1 Req 17:
Test Type: Capability

A.2.1.4 Geometry Columns

A.2.1.4.1 Data

A.2.1.4.1.1 Table Definition

Test Case ID: /opt/features/geometry_columns/data/table_def
Test Purpose: Verify that the gpkg_geometry_columns table exists and has the correct definition.
Test Method:

1) SELECT sql FROM sqlite_master WHERE type = 'table' AND tbl_name = 'gpkg_geometry_columns'
2) Fail if returns an empty result set.
3) Pass if the column names and column definitions in the returned Create TABLE statement in the sql column value, including data type, nullability, default values and primary, foreign and unique key constraints match all of those in the contents of C.4 Table 22. Column order, check constraint and trigger definitions, and other column definitions in the returned sql are irrelevant.
4) Fail otherwise.

Reference: Clause 2.1.5.1.1 Req 18:
Test Type: Basic

A.2.1.4.1.2 Table Data Values

Test Case ID: /opt/features/geometry_columns/data/data_values_table_name
Test Purpose: Verify that the table_name column values in the gpkg_geometry_columns table are valid.
Test Method:

1) SELECT DISTINCT table_name FROM gpkg_geometry columns
2) Not testable if returns an empty result set.
3) For each row from setp 1
   a. Fail if table_name value is NULL
4) SELECT DISTINCT ggc.table_name AS ggc_table, gc.table_name
   FROM gpkg_geometry_columns AS ggc LEFT OUTER JOIN geopackage_contents AS gc ON ggc.table_name = gc.table_name
5) For each row from step 4
   a. Fail if ggc.table_name != gc.table_name
6) Pass if no fails.

Reference: Clause 2.1.5.1.2 Req 19:
Test Type: Capability

Test Case ID: /opt/features/geometry_columns/data/data_values_column_name

Field Cod

Test Purpose: Verify that the column_name column values in the gpkg_geometry_columns table are valid.
Test Method:

1) SELECT table_name, column_name FROM gpkg_geometry_columns
2) Not testable if returns an empty result set
3) For each row from step 1
   a. PRAGMA table_info(table_name)
   b. Fail if gpkg_geometry_columns.column_name value does not equal a name column value returned by PRAGMA table_info.

Pass if no fails. Reference: Clause 2.1.5.1.2 Req 20:
Test Type: Capability

Test Case ID: /opt/features/geometry_columns/data/data_values_geometry_type_name
Test Purpose: Verrify that the geometry_type_name column values in the gpkg_geometry_columns table are valid.
Test Method:

1) SELECT DISTINCT geometry_type_name from gpkg_geometry_columns
2) Not testable if returns an empty result set
3) For each row from step 1
   a. Fail if a returned geometry_type value is not in Table 46 or Table 47 in Annex G
4) Pass if no fails.

Reference: Clause 2.1.5.1.2 Req 21:
Test Type: Capability

Test Case ID: /opt/features/geometry_columns/data/data_values_srs_id
Test Purpose: Verify that the gpkg_geometry_columns table srs_id column values are defined in the gpkg_spatial_ref_sys table.
Test Method:

1) SELECT srs_id FROM gpkg_geometry_columns
2) Not testable if returns an empty result set
3) SELECT srs_id FROM gpkg_geometry_columns WHERE srs_id NOT IN (SELECT srs_id FROM gpkg_spatial_ref_sys)
4) Fail if does not return an empty result set
5) Pass otherwise.

Reference: Clause 2.1.5.1.2 Req 22:
Test Type: Capability

Test Case ID: /opt/features/geometry_columns/data/data_values_z
Test Purpose: Verify that the gpkg_geometry_columns table z column values are valid.
Test Method:

Field Cod

1) SELECT z FROM gpkg_geometry_columns
2) Not testable if returns an empty result set
3) SELECT z FROM gpkg_geometry_columns WHERE z NOT IN (1,2,3)
4) Fail if does not return an empty result set
5) Pass otherwise.

Reference: Clause 2.1.5.1.2 Req 23:
Test Type: Capability

Test Case ID: /opt/features/geometry_columns/data/data_values_m
Test Purpose: Verify that the gpkg_geometry_columns table m column values are valid.
Test Method:

1) SELECT m FROM gpkg_geometry_columns
2) Not testable if returns an empty result set
3) SELECT m FROM gpkg_geometry_columns WHERE m NOT IN (1,2,3)
4) Fail if does not return an empty result set
5) Pass otherwise.

Reference: Clause 2.1.5.1.2 Req 24:
Test Type: **Capability**

A.2.1.5 Vector Features User Data Tables

A.2.1.5.1 Data

A.2.1.5.1.1 Table Definition

Test Case ID: /opt/features/vector_features/data/feature_table_integer_primary_key
Test Purpose: Verify that every vector features user data table has an integer primary key.
Test Method:

1) SELECT table_name FROM gpkg_contents WERE data_type = ‘features’
2) Not testable if returns an empty result set
3) For each row from step 1
   a. PRAGMA table_info(table_name)
   b. Fail if returns an empty result set
   c. Fail if result set does not contain one row where the pk column value is 1 and the not null column value is 1 and the type column value is “INTEGER”
4) Pass if no fails.

Reference: Clause 2.1.6.1.1 Req 25:
Test Type: Basic

Test Case ID: /opt/features/vector/features/data/feature_table_one_geometry_column
Test Purpose: Verify that every vector features user data table has one geometry column.

Test Method:

1) SELECT table_name FROM gpkg_contents WERE data_type = ‘features’
2) Not testable if returns an empty result set

Field Cod

3) For each row table name from step 1
    a. SELECT column_name from gpkg_geometry_columns where table_name = row table name
    b. Fail if returns more than one column name
4) Pass if no fails

Reference: **Clause 2.1.6.1.1 Req 26:**
Test Type: Capability

A.2.1.5.1.2 Table Data Values

Test Case ID: /opt/features/vector_features/data/data_values_geometry_type
Test Purpose: Verify that the geometry type of feature geometries are of the type or are assignable for the geometry type specified by the gpkg_geometry columns table geometry_type_name column value.
Test Method:

1) SELECT table_name AS tn, column_name AS cn, geometry_type_name AS gt_name FROM gpkg_geometry_columns WHERE table_name IN (SELECT table_name FROM gpkg_contents WHERE data_type = ‘features’)
2) Not testable if returns an empty result set
3) For each row from step 1
    a. SELECT DISTINCT ST_GeometryType(cn) FROM tn
    b. For each row actual_type_name from step a
        i. SELECT GPKG_IsAssignable(geometry_type_name, actual_type_name)
        ii. Fail if any returned 0
4) Pass if no fails

Reference: Clause 2.1.6.1.2 Req 27:
Test Type: Capability

Test Case ID: /opt/features/vector_features/data/tata_value_geometry_srs_id
Test Purpose: Verify the the srs_id of feature geometries are the srs_id specified for the gpkg_geometry_columns table srs_id column value.
Test Method:

1) SELECT table_name AS tn, column_name AS cn, srs_id AS gc_srs_id FROM gpkg_geometry_columns WHERE table_name IN (SELECT table_name FROM gpkg_contents where data_type = ‘features’)
2) Not testable if returns an empty result set
3) For each row from step 1
    a. SELECT DISTINCT st_srid(cn) FROM tn
    b. For each row from step a
        i. Fail if returnvalue not equal to gc_srs_id
4) Pass if no fails

Reference: Clause 2.1.6.1.2 Req 28:
Test Type: Capability

Field Cod

A.2.2 Tiles

A.2.2.1 Contents

A.2.2.1.1 Data

A.2.2.1.1.1 Contents Table –Tiles Row

Test Case ID: /opt/tiles/contents/data/tiles_row
Test Purpose: Verify that the gpkg_contents table_name value table exists and is apparently a tiles table for every row with a data_type column value of "tiles".
Test Method:

1) SELECT table_name FROM gpkg_contents WHERE data_type = "tiles"
2) Fail if returns empty result set
3) For each row from step 1
   a) PRAGMA table_info(table_name)
   b) Fail if returns an empty result set
   c) Fail if result set does not contain one row where the pk column value is 1 and the not null column value is 1 and the type column value is "INTEGER"and the name column value is "id"
   d) Fail if result set does not contain four other rows where the name column values are "zoom_level","tile_column","tile_row", and "tile_data".
4) Pass if no fails.

Reference: Clause 2.2.2.1.1 Req 29:
Test Type: Capability

A.2.2.2 Zoom Levels

A.2.2.2.1 Data

A.2.2.2.1.1 Zoom Times Two

Test Case ID: /opt/tiles/zoom_levels_data_zoom_times_two
Test Purpose: Verify that by default zoom level pixel sizes for tile matrix user data tables vary by powers of 2 between adjacent zoom levels in the tile matrix metadata table.
Test Method:

1) SELECT CASE
   WHEN (SELECT tbl_name FROM sqlite_master WHERE tbl_name = 'gpkg_extensions') = 'gpkg_extensions' THEN
   (SELECT table_name FROM geopackage_contents WHERE data_type = 'tiles' AND table_name NOT IN
   (SELECT table_name from gpkg_extensions WHERE extension_name = 'gpkg_zoom_other'))
   ELSE (SELECT table_name FROM geopackage_contents WHERE data_type = 'tiles')
   END;
2) Not testable if returns empty result set
3) For each row table_name from step 1
   a. SELECT zoom_level, pixel_x_size, pixel_y_size FROM tile_matrix_metadata WHERE table_name = selected table name ORDER BY zoom_level ASC

Field Cod

   b. Not testable if returns empty result set, or only one row
   c. Not testable if there are not two rows with adjacent zoom levels
   d. Fail if any pair of rows for adjacent zoom levels have pixel_x_size or pixel_y_size values that differ by other than powers of two
4) Pass if no fails

Reference: Clause 2.2.3.1.1 Req 30:
Test Type: Capability

A.2.2.3 Tile Encoding PNG

A.2.2.3.1 Data

A.2.2.3.1.1 MIME Type PNG

Test Case ID: /opt/tiles/tiles_encoding/data/mime_type_png
Test Purpose: Verify that a tile matrix user data table that contains tile data that is not MIME type image/jpeg by default contains tile data in MIME type image/png.
Test Method:

1) SELECT CASE
   WHEN (SELECT tbl_name FROM sqlite_master WHERE tbl_name = 'gpkg_extensions') = 'gpkg_extensions' THEN
   (SELECT table_name FROM geopackage_contents WHERE data_type = 'tiles' AND table_name NOT IN
    (SELECT table_name from gpkg_extensions WHERE extension_name IN ('gpkg_webp','gpkg_tiff','gpkg_nitf')))
   ELSE (SELECT table_name FROM geopackage_contents WHERE data_type = 'tiles')
   END;
2) Not testable if returns empty result set
3) For each row tbl_name from step 1
   a. SELECT tile_data FROM tbl_name
   b. For each row tile_data from step a
      i. Pass if tile data in MIME type image/png
4) Fail if no passes

Reference: Clause 2.2.4.1.1 Req 31:
Test Type: Capability

A.2.2.4 Tile Encoding JPEG

A.2.2.4.1 Data

A.2.2.4.1.1 MIME Type JPEG

Test Case ID: /opt/tiles/tiles_encoding/data/mime_type_jpeg
Test Purpose: Verify that a tile matrix user data table that contains tile data that is not MIME type image/png by default contains tile data in MIME type image/jpeg.
Test Method:

1) SELECT CASE

Field Cod

WHEN (SELECT tbl_name FROM sqlite_master WHERE tbl_name = 'gpkg_extensions') = 'gpkg_extensions' THEN
(SELECT table_name FROM geopackage_contents WHERE data_type = 'tiles' AND table_name NOT IN
(SELECT table_name from gpkg_extensions WHERE extension_name IN ('gpkg_webp','gpkg_tiff','gpkg_nitf')))
ELSE (SELECT table_name FROM geopackage_contents WHERE data_type = 'tiles')
END;

2) Not testable if returns empty result set
3) For each row tbl_name from step 1
   a. SELECT tile_data FROM tbl_name
   b. For each row tile_data from step a
      i. Pass if tile data in MIME type image/jpeg
4) Fail if no passes

Reference: Clause 2.2.5.1.1 Req 32:
Test Type: Capability

A.2.2.5 Tile Matrix Metadata

A.2.2.5.1 Data

A.2.2.5.1.1 Table Definition

Test Case ID: /opt/tiles/tile_matrix_metadata/data/table_def
Test Purpose: Verify that the gpkg_tile_matrix_metadata table exists and has the correct definition.
Test Method:

1) SELECT sql FROM sqlite_master WHERE type = 'table' AND tbl_name = 'gpkg_tile_matrix_metadata'
2) Fail if returns an empty result set.
3) Pass if the column names and column definitions in the returned CREATE TABLE statement in the sql column value,, including data type, nullability, default values and primary, foreign and unique key constraints match all of those in the contents of C.7 Table 25. Column order, check constraint and trigger definitions, and other column definitions in the returned sql are irrelevant.
4) Fail otherwise.

Reference: Clause 2.2.6.1.1 Req 33:
Test Type: Basic

A.2.2.5.1.2 Table Data Values

Test Case ID: /opt/tiles/tile_matrix_metadata/data_values_table_name
Test Purpose: Verify that values of the gpkg_tile_matrix_metadata table_name column reference values in the gpkg_contents table_name column for rows with a data type of “tiles”.
Test Method:

Field Cod

1) SELECT table_name FROM gpkg_tile_matrix_metadata
2) Not testable if returns an empty result set
3) SELECT table_name FROM gpkg_tile_matrix_metadata tmm WHERE table_name NOT IN (SELECT table_name FROM gpkg_contents gc WHERE tmm.table_name = gc.table_name)
4) Fail if result set contains any rows
5) Pass otherwise

Reference: Clause 2.2.6.1.2 Req 34:
Test Type: Capability

Test Case ID: /opt/tiles/tile_matrix_metadata/data/data_values_zoom_level_rows
Test Purpose: Verify that the gpkg_tile_matrix_metadata table contains a row record for each zoom level that contains one or more tiles in each tile matrix user data table.
Test Method:

1) SELECT table_name AS <user_data_tiles_table> from gpkg_contents where data_type = 'tiles'
2) Not testable if returns an empty result set
3) For each row from step 1
   a. SELECT DISTINCT gtmm.zoom_level AS gtmm_zoom, udt.zoom_level AS udtt_zoom FROM tile_matrix_metadata AS gtmm
      LEFT OUTER JOIN <user_data_tiles_table> AS udtt ON udtt.zoom_level = gtmm.zoom_level AND gtmm.t_table_name = '<user_data_tiles_table>'
   b. Fail if any gtmm_zoom column value in the result set is NULL
4) Pass if no fails

Reference: Clause 2.2.6.1.2 Req 35:
Test Type: Capability

Test Case ID: /opt/tiles/tile_matrix_metadata/data/data_values_zoom_level
Test Purpose: Verify that zoom level column values in the gpkg_tile_matrix_metadata table are not negative.
Test Method:

1) SELECT zoom_level FROM gpkg_tile_matrix_metadata
2) Not testable if returns an empty result set
3) SELECT min(zoom_level) FROM gpkg_tile_matrix_metadata.
4) Fail if less than 0.
5) Pass otherwise.

Reference: Clause 2.2.6.1.2 Req 36:
Test Type: Capability

Test Case ID: /opt/tiles/tile_matrix_metadata/data/data_values_matrix_width

Field Cod

Test Purpose: Verify that the matrix_width values in the gpkg_tile_matrix_metadata table are valid.
Test Method:

1) SELECT matrix_width FROM gpkg_tile_matrix_metadata
2) Not testable if returns an empty result set
3) SELECT min(matrix_width) FROM gpkg_tile_matrix_metadata.
4) Fail if less than 1.
5) Pass otherwise.

Reference: Clause 2.2.6.1.2 Req 37:
Test Type: Capabilty

Test Case ID: /opt/tiles/tile_matrix_metadata/data/data_values_matrix_height
Test Purpose: Verify that the matrix_height values in the gpkg_tile_matrix_metadata table are valid.
Test Method:

1) SELECT matrix_height FROM gpkg_tile_matrix_metadata
2) Not testable if returns an empty result set
3) SELECT min(matrix_height) FROM gpkg_tile_matrix_metadata.
4) Fail if less than 1.
5) Pass otherwise.

Reference: Clause 2.2.6.1.2 Req 38:
Test Type: Capability

Test Case ID: /opt/tiles/tile_matrix_metadata/data/data_values_tile_width
Test Purpose: Verify that the tile_width values in the gpkg_tile_matrix_metadata table are valid.
Test Method:

1) SELECT tile_width FROM gpkg_tile_matrix_metadata
2) Not testable if returns an empty result set
3) SELECT min(tile_width) FROM gpkg_tile_matrix_metadata.
4) Fail if less than 1.
5) Pass otherwise.

Reference: Clause 2.2.6.1.2 Req 39:
Test Type: Capability

Test Case ID: /opt/tiles/tile_matrix_metadata/data/data_values_tile_height
Test Purpose: Verify that the tile_height values in the gpkg_tile_matrix_metadata table are valid.
Test Method:

1) SELECT tile_height FROM gpkg_tile_matrix_metadata

Field Cod

2) Not testable if returns an empty result set
3) SELECT min(tile_height) FROM gpkg_tile_matrix_metadata.
4) Fail if less than 1.
5) Pass otherwise.

Reference: Clause 2.2.6.1.2 Req 40:
Test Type: Capability

Test Case ID: /opt/tiles/tile_matrix_metadata/data/data_values_pixel_x_size
Test Purpose: Verify that the pixel_x_size values in the gpkg_tile_matrix_metadata table are valid.
Test Method:

1) SELECT pixel_x_size FROM gpkg_tile_matrix_metadata
2) Not testable if returns an empty result set
3) SELECT min(pixel_x_size) FROM gpkg_tile_matrix_metadata.
4) Fail if less than 0.
5) Pass otherwise.

Reference: Clause 2.2.6.1.2 Req 41:
Test Type: Capability

Test Case ID: /opt/tiles/tile_matrix_metadata/data/data_values_pixel_y_size
Test Purpose: Verify that the pixel_y_size values in the gpkg_tile_matrix_metadata table are valid.
Test Method:

1) SELECT pixel_y_size FROM gpkg_tile_matrix_metadata
2) Not testable if returns an empty result set
3) SELECT min(pixel_y_size) FROM gpkg_tile_matrix_metadata.
4) Fail if less than 0.
5) Pass otherwise.

Reference: Clause 2.2.6.1.2 Req 42:
Test Type: Capability

Test Case ID: /opt/tiles/tile_matrix_metadata/data/data_values_pixel_size_sort
Test Purpose: Verify that the pixel_x_size and pixel_y_size column values for zoom level column values in a gpkg_tile_matrix_metadata table sorted in ascending order are sorted in descending order, showing that lower zoom levels are zoomed "out".
Test Method:

1) SELECT table_name FROM gpkg_contents WHERE data_type = 'tiles'
2) Not testable if returns empty result set
3) For each row table_name from step 1

Field Cod

   a. SELECT zoom_level, pixel_x_size, pixel_y_size from gpkg_tile_matrix_metadata WHERE table_name = row table name ORDER BY zoom_level ASC
   b. Not testable if returns empty result set
   c. Fail if pixel_x_sizes are not sorted in descending order
   d. Fail if pixel_y_sizes are not sorted in descending order
4) Pass if testable and no fails

Reference: Clause 2.2.6.1.2 Req 43:
Test Type: Capability

A.2.2.6 Tile Matrix User Data

A.2.2.6.1 Data

A.2.2.6.1.1 Table Definition

Test Case ID: /opt/tiles/tile_matrix/data/table_def
Test Purpose: Verify that multiple tile matrix sets are stored in different tiles tables with unique names containing the required columns.
Test Method:

1) SELECT COUNT(table_name) FROM gpkg_contents WERE data_type = “tiles”
2) Fail if less than 2
3) SELECT table_name FROM gpkg_contents WHERE data_type = “tiles”
4) For each row from step 3
   a. PRAGMA table_info(table_name)
   b. Fail if returns an empty result set
   c. Fail if result set does not contain one row where the pk column value is 1 and the not null column value is 1 and the type column value is “INTEGER”and the name column value is “id”
   d. Fail if result set does not contain four other rows where the name column values are “zoom_level”,”tile_column”,”tile_row”, and “tile_data”.
5) Pass if no fails

Reference: Clause 2.2.7.1.1 Req 44:
Test Type: Basic

A.2.2.6.1.2 Table Data Values

Test Case ID: /opt/tiles/tile_matrix/data/data_values_zoom_levels
Test Purpose: Verify that the zoom level column values in each tile matrix user data table are within the range of zoom levels defined by rows in the tile_matrix_metadata table.
Test Method:

1) SELECT DISTINCT table_name AS <user_data_tiles_table> FROM gpkg_tile_matrix_metadata
2) Not testable if returns an empty result set
3) For each row <user_data_tiles_table> from step 1
   a. SELECT zoom_level FROM <user_data_tiles_table>
   b. If result set not empty
      i. SELECT MIN(gtmm.zoom_level) AS min_gtmm_zoom, MAX(gtmm.zoom_level) AS max_gtmm_zoom FROM

Field Cod

gpkg_tile_matrix_metadata WHERE table_name = <user_data_tiles_table>

ii. SELECT id FROM <user_data_tiles_table> WHERE zoom_level < min_gtmm_zoom
iii. Fail if result set not empty
iv. SELECT id FROM <user_data_tiles_table> WHERE zoom_level > max_gtmm_zoom
v. Fail if result set not empty
vi. Log pass otherwise

4) Pass if logged pas and no fails

Reference: Clause 2.2.7.1.2 Req 45:
Test Type: Capability

Test Case ID: /opt/tiles/tile_matrix/data/data_values_tile_column
Test Purpose: Verify that the tile_column column values for each zoom level value in each tile matrix user data table are within the range of columns defined by rows in the tile_matrix_metadata table.
Test Method:

1) SELECT DISTINCT table_name AS <user_data_tiles_table> FROM gpkg_tile_matrix_metadata
2) Not testable if returns an empty result set
3) For each row <user_data_tiles_table> from step 1
   a. SELECT DISTINCT gtmm.zoom_level AS gtmm_zoom, gtmm.matrix_width AS gtmm_width, udt.zoom_level AS udt_zoom, udt.tile_column AS udt_column FROM tile_matrix_metadata AS gtmm LEFT OUTER JOIN <user_data_tiles_table> AS udt ON udt.zoom_level = gtmm.zoom_level AND gtmm.t_table_name = '<user_data_tiles_table>' AND (udt_column < 0 OR udt_column > (gtmm_width - 1))
   b. Fail if any udt_column value in the result set is not NULL
   c. Log pass otherwise
4) Pass if logged pass and no fails

Reference: Clause 2.2.7.1.2 Req 46:
Test Type: Capability

Test Case ID: /opt/tiles/tile_matrix_data/data_values_tile_row
Test Purpose: Verify that the tile_row column values for each zoom level value in each tile matrix user data table are within the range of rows defined by rows in the tile_matrix_metadata table.
Test Method:

1) SELECT DISTINCT table_name AS <user_data_tiles_table> FROM gpkg_tile_matrix_metadata
2) Not testable if returns an empty result set
3) For each row <user_data_tiles_table> from step 1

Field Cod

   a. SELECT DISTINCT gtmm.zoom_level AS gtmm_zoom, gtmm.matrix_height AS gtmm_height, udt.zoom_level AS udt_zoom, udt.tile_row AS udt_row FROM tile_matrix_metadata AS gtmm LEFT OUTER JOIN <user_data_tiles_table> AS udt ON udt.zoom_level = gtmm.zoom_level AND gtmm.t_table_name = '<user_data_tiles_table> ' AND (udt_row < 0 OR udt_row > (gtmm_height - 1))
   b. Fail if any udt_row value in the result set is not NULL
   c. Log pass otherwise
4) Pass if logged pass and no fails

Reference: Clause 2.2.7.1.2 Req 47:
Test Type: Capability

A.2.3 Schema

A.2.3.1 Data Columns

A.2.3.1.1 Data

A.2.3.1.1.1 Table Definition

Test Case ID: /opt/schema/data_columns/data_table_def
Test Purpose: Verify that the gpkg_data_columns table exists and has the correct definition.
Test Method:

1) SELECT sql FROM sqlite_master WHERE type = 'table' AND tbl_name = 'gpkg_data_columns'
2) Fail if returns an empty result set
3) Pass if column names and column definitions in the returned CREATE TABLE statement in the sql column value, including data type, nullability, default values and primary, foreign and unique key constraints match all of those in the contents of C.1 Table 32. Column order, check constraint and trigger definitions, and other column definitions in the returned sql are irrelevant.
4) Fail otherwise.

Reference: Clause 2.3.1.1.1 Req 48:
Test Type: Basic

A.2.3.1.1.2 Data Values

Test Case ID: /opt/schema/data_columns/data/data_values_table_name
Test Purpose: Verify that values of the gpkg_data_columns table_name column reference values in the gpkg_contents table_name column.
Test Method:

1) SELECT table_name FROM gpkg_data columns
2) Not testable if returns an empty result set
3) SELECT table_name FROM gpkg_data_columns gdc WHERE table_name NOT IN (SELECT table_name FROM gpkg_contents gc WHERE gdc.table_name = gc.t_table_name)
4) Fail if result set contains any rows
5) Pass otherwise

Reference: Clause 2.3.1.1.2 Req 49:

Field Cod

Test Type: Capability

Test Case ID: /opt/schema/data_columns/data/data_values_column_name
Test Purpose: Verify that for each gpkg_data_columns row, the column_name value is the name of a column in the table_name table.
Test Method:

1) SELECT table_name, column_name FROM gpkg_data_columns
2) Not testable if returns an empty result set
3) For each row from step 1
   a. PRAGMA table_info(table_name)
   b. Fail if gpkg_data_columns.column_name value does not equal a name column value returned by PRAGMA table_info.
4) Pass if no fails.

Reference: Clause 2.3.1.1.2 Req 50:
Test Type: Capability

A.2.4 Metadata

A.2.4.1 Metadata Table

A.2.4.1.1 Data

A.2.4.1.1.1 Table Definition

Test Case ID: /opt/metadata/metadata/data/table_def
Test Purpose: Verify that the gpkg_metadata table exists and has the correct definition.
Test Method:

1) SELECT sql FROM sqlite_master WHERE type = 'table' AND tbl_name = 'gpkg_metadata'
2) Fail if returns an empty result set.
3) Pass if the column names and column definitions in the returned Create TABLE statement in the sql column value, including data type, nullability, default values and primary, foreign and unique key constraints match all of those in the contents of Table 33. Column order, check constraint and trigger definitions, and other column definitions in the returned sql are irrelevant.
4) Fail otherwise.

Reference: Clause 2.4.2.1.1 Req 51:
Test Type: Basic

A.2.4.1.1.2 Table Data Values

Test Case ID: /opt/metadata/metadata/data/data_values_md_scope
Test Purpose: Verify that each of the md_scope column values in a gpkg_metadata table is one of the name column values from Table 11 in clause 2.4.2.1.2.
Test Method:

1) SELECT md_scope FROM gpkg_metadata

Field Cod

2) Not testable if returns an empty result set
3) For each row returned from step 1
    a. Fail if md_scope value not one of the name column values from Table 11 in clause 2.4.2.1.2
4) Pass if no fails

Reference: Clause 2.4.2.1.2 Req 52:
Test Type: Capabilities

A.2.4.2 Metadata Reference Table

A.2.4.2.1 Data

A.2.4.2.1.1 Table Definition

Test Case ID: /opt/metadata/metadata_reference_data_table_def
Test Purpose: Verify that the gpkg_metadata_reference table exists and has the correct definition.
Test Method:

1) SELECT sql FROM sqlite_master WHERE type = 'table' AND tbl_name = 'gpkg_metadata_reference'
2) Fail if returns an empty result set.
3) Pass if the column names and column definitions in the returned Create TABLE statement in the sql column value, including data type, nullability, default values and primary, foreign and unique key constraints match all of those in the contents of Table 34. Column order, check constraint and trigger definitions, and other column definitions in the returned sql are irrelevant.
4) Fail otherwise.

Reference: Clause 2.4.3.1.1 Req 53:
Test Type: Basic

A.2.4.2.1.2 Data Values

Test Case ID: /opt/metadata/metadata_reference/data/data_values_reference_scope
Test Purpose: Verify that gpkg_metadata_reference table reference_scope column values are valid.
Test Method:

1) SELECT reference_scope FROM gpkg_metadata_reference
2) Not testable if returns an empty result set
3) SELECT reference_scope FROM gpkg_metadata_reference WHERE reference_scope NOT IN (‘geopackage’,’table’,’column’,’row’,’row/col’)
4) Fail if does not return an empty result set
5) Pass otherwise.

Reference: Clause 2.4.3.1.2 Req 54:
Test Type: Capability

Test Case ID: /opt/metadata/metadata_reference/data/data_values_table_name

Field Cod

Test Purpose: Verify that gpkg_metadata_reference table_name column values are NULL for rows with reference_scope values of ‘geopackage’, and reference gpkg_contents table_name values for all other reference_scope values.
Test Method:

1) SELECT table_name FROM gpkg_metadata_reference
2) Not testable if returns an empty result set
3) SELECT table_name FROM gpkg_metadata_reference WHERE reference_scope = 'geopackage'
4) Fail if result set contains any non-NULL values
5) SELECT table_name FROM metadata_reference WHERE reference_scope != 'geopackage' AND table_name NOT IN (SELECT table_name FROM gpkg_contents)
6) Fail if result set is not empty
7) Pass otherwise.

Reference: Clause 2.4.3.1.2 Req 55:
Test Type: Capability

Test Case ID: /opt/metadata/metadata_reference/data/data_values_column_name
Test Purpose: Verify that gpkg_metadata_reference column_name column values are NULL for rows with reference scope values of ‘geopackage’, ‘table’, or ‘row’, and contain the name of a column in table_name table for other reference scope values.
Test Method:

1) SELECT column_name FROM gpkg_metadata_reference
2) Not testable if returns an empty result set
3) SELECT column_name FROM gpkg_metadata_reference WHERE reference_scope IN ('geopackage', ‘table’, ‘row’)
4) Fail if result set contains any non-NULL values
5) SELECT <table_name>, <column_name> FROM metadata_reference WHERE reference_scope NOT IN ('geopackage', ‘table’, ‘row’)
6) For each row from step 5
   a. SELECT sql FROM sqlite_master WHERE type = 'table' AND tbl_name = '<table_name>'
   b. Fail if returns an empty result set.
   c. Fail if the one of the column names in the returned sql Create TABLE statement is not <column_name>
   d. Log pass otherwise
7) Pass if logged pass and no fails.

Reference: Clause 2.4.3.1.2 Req 56:
Test Type: Capability

Test Case ID: /opt/metadata/metadata_reference/data/data_values_row_id_value

Field Cod

Test Purpose: Verify that gpkg_metadata_reference row_id_value column values are NULL for rows with reference scope values of ‘geopackage’, ‘table’, or ‘row’, and contain the ROWID of a row in the table_name for other reference scope values.
Test Method:

1) SELECT row_id_value FROM gpkg_metadata_reference
2) Not testable if returns an empty result set
3) SELECT row_id_value FROM gpkg_metadata_reference WHERE reference_scope IN ('geopackage', ‘table’, ‘row’)
4) Fail if result set contains any non-NULL values
5) For each SELECT <table_name>, <row_id_value> FROM gpkg_metadata_reference WHERE reference_scope NOT IN ('geopackage', ‘table’, ‘row’)
6) For each row from step 5
   a. SELECT * FROM <table_name> WHERE ROWID = <row_id_value>
   b. Fail if result set is empty
   c. Log pass otherwise
7) Pass if logged pass and no fails.

Reference: Clause 2.4.3.1.2 Req 57:
Test Type: Capability

Test Case ID: /opt/metadata/metadata_reference/data/data_values_timestamp
Test Purpose: Verify that every gpkg_metadata_reference table row timestamp column value is in ISO 8601 UTC format.
Test Method:

1) SELECT timestamp from gpkg_metadata_reference.
2) Not testable if returns an empty result set
3) For each row from step 1
   a. Fail if format of returned value does not match yyyy-mm-ddThh:mm:ss.hhhZ
   b. Log pass otherwise
4) Pass if logged pass and no fails.

Reference: Clause 2.4.3.1.2 Req 58:
Test Type: Capability

Test Case ID: /opt/metadata/metadata_reference/data/data_values_md_file_id
Test Purpose: Verify that every gpkg_metadata_reference table row md_file_id column value is an id column value from the gpkg_metadata table.
Test Method:

1) SELECT md_file_id FROM gpkg_metadata_reference
2) Not testable if returns an empty result set
3) SELECT gmr.md_file_id, gm.id FROM gpkg_metadata_reference AS gmr
4) LEFT OUTER JOIN gpkg_metadata gm ON gmr.md_file_id = gm.id
5) Fail if result set is empty
6) Fail if any result set gm.id values are NULL

Field Cod

7) Pass otherwise

Reference: Clause 2.4.3.1.2 Req 59:
Test Type: Capability

Test Case ID: /opt/metadata/metadata_reference/data/data_values_md_parent_id
Test Purpose: Verify that every gpkg_metadata_reference table row md_parent_id column value that is not null is an id column value from the gpkg_metadata_table that is not equal to the md_file_id column value for that row.
Test Method:

1) SELECT md_file_id FROM gpkg_metadata_reference
2) Not testable if returns an empty result set
3) SELECT gmr.md_file_id, gmr.md_parent_id
4) FROM gpkg_metadata_reference AS gmr
5) WHERE gmr.md_file_id == gmr.md_parent_id
6) Fail if result set is not empty
7) SELECT gmr.md_file_id, gmr.md_parent_id, gm.id
8) FROM gpkg_metadata_reference AS gmr
9) LEFT OUTER JOIN gpkg_metadata gm ON gmr.md_parent_id =gm.id
10) Fail if any result set gm.id values are NULL
11) Pass otherwise

Reference: Clause 2.4.3.1.2 Req 60:
Test Type: Capability

A.2.5 Extension Mechanism

A.2.5.1 Extensions

A.2.5.1.1 Data

A.2.5.1.1.1 Table Definition

Test Case ID: /opt/extension_mechanism/extensions/data/table_def
Test Purpose: Verify that a gpkg_extensions table exists and has the correct definition.
Test Method:

1) SELECT sql FROM sqlite_master WHERE type = 'table' AND tbl_name = 'gpkg_extensions'
2) Fail if returns an empty result set.
3) Pass if the column names and column definitions in the returned Create TABLE statement in the sql column value, including data type, nullability, default values and primary, foreign and unique key constraints match all of those in the contents of Table 23. Column order, check constraint and trigger definitions, and other column definitions in the returned sql are irrelevant.
4) Fail otherwise.

Reference: Clause 2.6.1.1.1 Req 61:
Test Type: Basic

Field Cod

A.2.5.1.1.2 Table Data Values

Test Case ID: /opt/extension_metchanism/extensions/data/data_values_table_name
Test Purpose: Verify that the table_name column values in the gpkg_extensions table are valid.
Test Method:

1) SELECT table_name, column_name FROM gpkg_extensions
2) Not testable if returns an empty result set
3) For each row from step one
   a. Fail if table_name value is NULL and column_name value is not NULL.
   b. SELECT DISTINCT ge.table_name AS ge_table, sm.tbl_name
      FROM gpkg_extensions AS ge LEFT OUTER JOIN sqlite_master AS sm ON ge.table_name = sm.tbl_name
   c. Log pass if every row ge.table_name = sm.tbl_name (MAY both be NULL).
4) Pass if logged pass and no fails.

Reference: Clause 2.6.1.1.2 Req 63:
Test Type: Capability

Test Case ID: /opt/extension_metchanism/extensions/data/data_values_column_name
Test Purpose: Verify that the column_name column values in the gpkg_extensions table are valid.
Test Method:

1) SELECT table_name, column_name FROM gpkg_extensions
2) Not testable if returns an empty result set
3) SELECT table_name, column_name FROM gpkg_extensions WHERE table_name IS NOT NULL AND column_name IS NOT NULL
4) Pass if returns an empty result set
5) For each row from step 3
   a. PRAGMA table_info(table_name)
   b. Fail if gpkg_extensions.column_name value does not equal a name column value returned by PRAGMA table_info.
   c. Log pass otherwise
6) Pass if logged pass and no fails.

Reference: Clause 2.6.1.1.2 Req 63:
Test Type: Capability

Test Case ID: /opt/extension_mechanism/extensions/data/data_values_extension_name
Test Purpose: Verify that the extension_name column values in the gpkg_extensions table are valid.
Test Method:

1) SELECT extension_name FROM gpkg_extensions
2) Not testable if returns an empty result set

Field Cod

3) For each row returned from step 1
    a. Log pass if extension_name is one of those listed in Table 14
    b. Separate extension_name into <author> and <extension> at the first "_"
    c. Fail if <author> is "gpkg"
    d. Fail if <author> contains characters other than [a-zA-Z0-9]
    e. Fail if <extension> contains characters other than [a-zA-Z0-9_]
    f. Log pass otherwise
4) Pass if logged pass and no fails.

Reference: Clause 2.6.1.1.2 Req 64:
Test Type: Capability

A.2.5.1.2 API

A.2.5.1.2.1 API GeoPackage SQLite Config

Test Case ID: /opt/extension_mechanism/extensions/api/api_geopackage_sqlite_config
Test Purpose: Verify that a GeoPackage SQLite Extension has the API GeoPackage SQLite Configuration compile and run time options.
Test Method:

1) SELECT sqlite_compileoption_used**(**'SQLITE_OMIT_LOAD_EXTENSION'**)**
2) Fail if returns 1
3) SELECT sqlite_compileoption_used**(**'SQLITE_OMIT_VIRTUALTABLE'**)**
4) Fail if returns 1
5) SELECT sqlite_compileoption_used**(**'SQLITE_ENABLE_RTREE_'**)**
6) Fail if returns 0
7) SELECT sqlite_compileoption_used**(‘**SQLITE_RTREE_INT_ONLY’)
8) Fail if returns 1
9) Pass otherwise

Reference: Clause 2.6.1.2.1 Req 65:
Test Type: Basic

A.2.5.1.2.2 Safe GeoPackage SQLite Config

Test Case ID: /opt/extension_mechanism/extensions/api/safe_geopackage_sqlite_config
Test Purpose: Verify that a GeoPackage SQLite Extension has the Safe GeoPackage SQLite Configuration compile and run time options.
Test Method:

1) SELECT sqlite_compileoption_used**(**‘SQLITE_DEFAULT_FOREIGN_KEYS '**)**
2) Fail if returns 0
3) SELECT sqlite_compileoption_used('SQLITE_OMIT_FOREIGN_KEY')
4) Fail if returns 1
5) PRAGMA foreign_keys
6) Fail if returns 0
7) SELECT sqlite_compileoption_used('SQLITE_OMIT_INTEGRITY_CHECK')
8) Fail if returns 1

Field Cod

9) SELECT sqlite_compileoption_used('SQLITE_OMIT_SUBQUERY')
10) Fail if returns 1
11) SELECT sqlite_compileoption_used('SQLITE_OMIT_TRIGGER')
12) Fail if returns 1
13) Pass otherwise

Reference: Clause 2.6.1.2.2 Req 66:
Test Type: Basic

A.3 Registered Extensions

Test Case ID: /reg_ext/all/author_name/not_gpkg/not_features_or_tiles
Test Purpose: Verify that any table in a GeoPackage file subject to a registered extension with an author_name other than "gpkg" is not described by a gpkg_contents table row with a data_type value of 'features' or 'tiles'.
Test Method:

1) /opt/extension_mechanism/extensions/data/table_def
2) Not testable if failed
3) SELECT table_name FROM geopackage_contents
   WHERE data_type IN ('features','tiles') AND table_name IN
   (SELECT table_name FROM gpkg_extensions WHERE
   substr(lower(extension_name),1,4) != 'gpkg')
4) Fail if result set is not empty
5) Pass otherwise

Reference: Clause 3 Req 67:
Test Type: Basic

A.3.1 Features

A.3.1.1 Geometry Encoding

A.3.1.1.1 Data

A.3.1.1.1.1 BLOB Format – Extensions Name

Test Case ID: /reg_ext/features/geometry_encoding/data/ext_name
Test Purpose: Verify that an extension name in the form <author_name>_geometry_encoding is defined for an author name other than "gpkg" for each geometry BLOB format other than GeoPackageBinary used in a GeoPackage file.
Test Method:

1) SELECT table_name, column_name FROM gpkg_geometry_columns WHERE table_name IN (SELECT table_name FROM gpkg_contents WHERE data_type NOT IN ('features', 'tiles'))
2) Not testable if returns an empty result set
3) For each row table_name, column_name from step 1
    a. SELECT result_set_column_name FROM result_set_table_name
    b. Not testable if returns an empty result set

Field Cod

```
c. For each geometry column value from step a
    i. If the first three bytes of geometry column value are not "GPB", then
        1. /opt/extension_mechanism/extensions/data/table_def
        2. Fail if failed
        3. SELECT extension_name FROM gpkg_extensions WERE
           table_name = result_set_table_name AND column_name =
           result_set_column_name AND
           substr(lower(extension_name),1,4) != 'gpkg')
            a. Fail if returns an empty result set
            b. Separate extension_name into <author> and
               <extension> at the first "_"
            c. Fail if <extension> is not '_geometry_encoding'.
            d. Otherwise log pass
4) Pass if logged pass and no fails
```

Reference: Clause 3.1.1.1.2 Req 68:
Test Type: Basic

A.3.1.1.1.2 BLOB Format – Extensions Row

Test Case ID: /reg_ext/features/geometry_encoding/data/ext_row
Test Purpose: Verify that the gpkg_extensions table contains a row with an extension_name in the form <author_name>_geometry_encoding is defined for an author name other than "gpkg" for each table_name and column_name that contain a geometry BLOB format other than GeoPackageBinary in a GeoPackage file.
Test Method:

Same as /reg_ext/features/geometry_encoding/data/ext_name

Reference: Clause 3.1.1.1.3 Req 69:
Test Type: Capability

A.3.1.2 Geometry Types

A.3.1.2.1 Data

A.3.1.2.1.1 Extension Types

Test Case ID:
/reg_ext/features/geometry_encoding/data/extension_types_existing_sparse_data
Test Purpose: Verify that existing extended non-linear geometry types are stored in valid GeoPackageBinary format encodings.
Test Method:

```
1) SELECT table_name FROM gpkg_geometry_columns
2) Not testable if returns an empty result set
3) SELECT table_name AS tn, column_name AS cn FROM gpkg_geometry_columns
   WHERE table_name IN (SELECT table_name FROM gpkg_contents WHERE
   data_type = 'features'),
4) Fail if returns an empty result set
5) For each row from step 3
```

Field Cod

   a. SELECT cn FROM tn;
   b. For each row from step a, if bytes 2-5 of cn.wkb as uint32 in endianness of gc.wkb byte 1of cn from #1 are a geometry type value from Annex G Table 46, then
      i. Log cn.header values, wkb endianness and geometry type
      ii. If cn.wkb is not correctly encoded per ISO 13249-3 clause 5.1.46 then log fail
      iii. If cn.flags.E is 1 - 4 and some cn.wkbx is outside of cn.envelope.minx,maxx then log fail
      iv. If cn.flags.E is 1 - 4 and some gc.wkby is outside of cn.envelope.miny,maxy then log fail
      v. If cn.flags.E is 2,4 and some gc.wkb.z is outside of cnenvelope.minz,maxz then log fail
      vi. If cn.flags.E is 3,4 and some gc.wkb.m is outside of cn.envelope.minm,maxm then log fail
      vii. If cn.flags.E is 5-7 then log fail
      viii. Otherwise log pass
6) Log pass if log contains pass and no fails

Reference: **Clause 3.1.2.1.1 Req 70:**
Test Type: **Capability**

Test Case ID:
/reg_ext/features/geometry_encoding/data/extension_types_all_types_test_data
Test Purpose: Verify that all extended non-linear geometry types and options are stored in valid GeoPackageBinary format encodings.
Test Method:

1) Open GeoPackage that has feature geometry values of geometry type in Annex G, for an assortment of srs_ids, for an assortment of coordinate values, without and with z and / or m values, in both big and little endian encodings:
2) /reg_ext/features/geometry_encoding/data/extension_types_existing_sparse_data
3) Pass if log contains pass record for big and little endian GPB headers containing big and little endian WKBs for 0-1 envelope contents indicator codes for every geometry type value from Annex G Table 47 without and with z and/or m values.
4) Fail otherwise

Reference: Clause 3.1.2.1.1 Req 70:
Test Type: Capability

A.3.1.2.1.2 Geometry Types -- Extensions Name

Test Case ID: /reg_ext/features/geometry_encoding/data/extension_name
Test Purpose: Verify that an extension name in the form gpkg_geom_<gname> is defined for each <gname> extension geometry type from Annex G **Table 47** used in a GeoPackage file.
Test Method:

Field Cod

```
1) SELECT table_name, column_name FROM gpkg_geometry_columns WHERE
   table_name IN (SELECT table_name FROM gpkg_contents WHERE data_type  ==
   'features'))
2) Not testable if result set is empty
3) For each row result set table_name, column_name from step 3
      a. SELECT result_set_column_name FROM result_set_table_name
      b. For each geometry column value from step a
         i.   If the first three bytes of each geometry column value are "GPB", then
                 1. /opt/extension_mechanism/extensions/data/table_def
                 2. Fail if failed
                 3. SELECT ST_GeometryType(geometry column value) AS
                    <gtype>;
                 4. SELECT extension_name FROM gpkg_extensions WERE
                    table_name = result_set_table_name AND column_name =
                    result_set_column_name AND extension_name = 'gpkg_geom_'
                    || <gtype>
                       a. Fail if result set is empty
                       b. Log pass otherwise
4) Pass if logged pass and no fails
```

Reference: Clause 3.1.2.1.2 Req 71:
Test Type: Basic

A.3.1.2.1.3 Geometry Types -- Extensions Row

Test Case ID: /reg_ext/features/geometry_encoding/data/extension_row
Test Purpose: Verify that the gpkg_extensions table contains a row with an extension_name in the form gpkg_geom_<gname> for each table_name and column_name in the gpkg_geometry_columns table with a <gname> geometry_type_name.
Test Method:

/reg_ext/features/geometry_encoding/data/extension_name

Reference: Clause 3.1.2.1.3 Req 72:
Test Type: Capability

A.3.1.3 Spatial Indexes

A.3.1.3.1 Data

A.3.1.3.1.1 Spatial Indexes Implementation

Test Case ID: /reg_ext/features/spatial_indexes/implementation
Test Purpose: Verify the correct implementation of spatial indexes on feature table geometry columns.
Test Method:

```
1) SELECT table_name, column_name FROM gpkg_geometry_columns WHERE
   table_name IN (SELECT table_name FROM gpkg_contents WHERE data_type  ==
   'features'))
2) Not testable if result set is empty
```

Field Cod

3) For each row table_name, column_name from step 1
    a. SELECT sql FROM sqlite_master WHERE tbl_name = 'rtree_' || result_set_table_name || '_' || result_set_column_name
    b. Not testable if result set is empty
    c. Fail if returned sql != 'CREATE VIRTUAL TABLE rtree_' ' || result_set_table_name || '_' || result_set_column_name || USING rtree(id, minx, maxx, miny, maxy)
    d. SELECT sql FROM sqlite_master WHERE type = 'trigger' AND tname = 'rtree_' || result_set_table_name || '_' || result_set_column_name || '_insert'
    e. Fail if returned sql != result of populating insert triggers template in Table 39 using result_set_table_name for <t> and result_set_column_name for <c>
    f. SELECT sql FROM sqlite_master WHERE type = 'trigger' AND name LIKE 'rtree_' || result_set_table_name || '_' || result_set_column_name || '_update%'
    g. Fail if returned sql != result of populating 4 update triggers templates in Table 39 using result_set_table_name for <t> and result_set_column_name for <c>
    h. SELECT sql FROM sqlite_master WHERE type='trigger' AND name = 'rtree_' || result_set_table_name || '_' || result_set_column_name || '_delete'
    i. Fail if returned sql != result of populating delete trigger template in Table 39 using result_set_table_name for <t> and result_set_column_name for <c>
    j. Log pass otherwise
4) Pass if logged pass and no fails

Reference: Clause 3.1.3.1.1 Req 73:
Test Type: Capability

A.3.1.3.1.2 Spatial Indexes – Extensions Name

Test Case ID: /reg_ext/features/spatial_indexes/extension_name
Test Purpose: Verify that the "gpkg_rtree_index" extension name is used to register spatial index extensions.
Test Method:

1) SELECT table_name, column_name FROM gpkg_geometry_columns WHERE table_name IN (SELECT table_name FROM gpkg_contents WHERE data_type == 'features'))
2) Not testable if result set is empty
3) For each row table_name, column_name from step 3
    a. SELECT sql FROM sqlite_master WHERE tbl_name = 'rtree_' || result_set_table_name || '_' || result_set_column_name
    b. Not testable if returns an empty result set
    c. /opt/extension_mechanism/extensions/data/table_def
    d. Fail if failed
    e. SELECT extension_name from gpkg_extensions WHERE table_name = result_set_table_name AND column_name = result_set_column_name
    f. Log pass if result is "gpkg_rtree_index"
    g. Fail otherwise
4) Pass if logged pass and no fails

Reference: Clause 3.1.3.1.2 Req 74:

Field Cod

Test Type: Basic

A.3.1.3.1.3 Spatial Indexes – Extensions Row

Test Case ID: /reg_ext/features/spatial_indexes/extension_row
Test Purpose: Verify that spatial index extensions are registered using the "gpkg_rtree_index" name in the gpkg_extensions table.
Test Method:

/reg_ext/features/spatial_indexes/extension_name

Reference: Clause 3.1.3.1.3 Req 75:
Test Type: Capability

A.3.1.4 Geometry Type Triggers

A.3.1.4.1 Data

A.3.1.4.1.1 Geometry Type Triggers Implementation

Test Case ID: /reg_ext/features/geometry_type_triggers/implementation
Test Purpose: Verify that user feature data table geometry type triggers are implemented correctly.
Test Method:

1) SELECT table_name, column_name FROM gpkg_geometry_columns WHERE table_name IN (SELECT table_name FROM gpkg_contents WHERE data_type == 'features'))
2) Not testable if returns an empty result set
3) For each row table_name, column_name from step 1
    a. SELECT sql FROM sqlite_master WHERE type = 'trigger' AND tbl_name = 'fgti_' || result_set_table_name || '_' || result_set_column_name
    b. Not testable if returns an empty result set
    c. Fail if sql != result of populating the first trigger template in Table 17 with <t> as result_set_table_name and <c> as result_set_column_name
    d. SELECT sql FROM sqlite_master WHERE type = 'trigger' AND tbl_name = 'fgtu_' || result_set_table_name || '_' || result_set_column_name
    e. Fail if sql != result of populating the second trigger template in Table 17 with <t> as result_set_table_name and <c> as result_set_column_name
    f. Log pass otherwise
4) Pass if logged pass and no fails

Reference: Clause 3.1.4.1.1 Req 76:
Test Type: Capability

A.3.1.4.1.2 Geometry Type Triggers – Extensions Name

Test Case ID: /reg_ext/features/geometry_type_triggers/extension_name
Test Purpose: Verify that the "gpkg_geometry_type_trigger" extension name is used to register geometry type triggers.
Test Method:

Field Cod

1) SELECT table_name, column_name FROM gpkg_geometry_columns WHERE table_name IN (SELECT table_name FROM gpkg_contents WHERE data_type == ‘features'))
2) Not testable if result set is empty
3) For each row table_name, column_name from step 1
    a. SELECT sql FROM sqlite_master WHERE type = ‘trigger’ AND tbl_name = ‘fgti_’ || result_set_table_name || ‘_’ || result_set_column_name
    b. Not testable if result set is empty
    c. /opt/extension_mechanism/extensions/data/table_def
    d. Fail if failed
    e. SELECT extension_name from gpkg_extensions WHERE table_name = result_set_table_name AND column_name = result_set_column_name
    f. Log pass if result is “gpkg_geometry_type_trigger”
    g. Fail otherwise
4) Pass if logged pass and no fails

Reference: Clause 3.1.4.1.2 Req 77:
Test Type: Basic

A.3.1.4.1.3 Geometry Type Triggers – Extensions Row

Test Case ID: /reg_ext/features/geometry_type_triggers/extension_row
Test Purpose: Verify that geometry type triggers are registered using the “gpkg_geometry_type_trigger” extension name.
Test Method:

/reg_ext/features/geometry_type_triggers/extension_name

Reference: Clause 3.1.4.1.3 Req 78:
Test Type: Capability

A.3.1.5 SRS_ID Triggers

A.3.1.5.1 Data

A.3.1.5.1.1 SRS_ID Triggers – Implementation

Test Case ID: /reg_ext/features/srs_id_triggers/implementation
Test Purpose: Verify that user feature data table srs_id triggers are implemented correctly.
Test Method:

1) SELECT table_name, column_name FROM gpkg_geometry_columns WHERE table_name IN (SELECT table_name FROM gpkg_contents WHERE data_type == ‘features'))
2) Not testable if result set is empty
3) For each row table_name, column_name from step 1
    a. SELECT sql FROM sqlite_master WHERE type = ‘trigger’ AND tbl_name = ‘fgsi_’ || result_set_table_name || ‘_’ || result_set_column_name
    b. Not testable if result set is empty

Field Cod

   c. Fail if sql != result of populating the first trigger template in Table 18 with <t> as result_set_table_name and <c> as result_set_column_name
   d. SELECT sql FROM sqlite_master WHERE type = ‘trigger’ AND tbl_name = ‘fgsu_’ || result_set_table_name || ‘_’ || result_set_column_name
   e. Fail if sql != result of populating the second trigger template in Table 18 with <t> as result_set_table_name and <c> as result_set_column_name
   f. Log pass otherwise
4) Pass if logged pass and no fails

Reference: Clause 3.1.5.1.1 Req 79:
Test Type: Capability

A.3.1.5.1.2 SRS_ID Triggers – Extensions Name

Test Case ID: /reg_ext/features/srs_id_triggers/extension_name
Test Purpose: Verify that the “gpkg_srs_id_trigger” extension name is used to register srs_id triggers.
Test Method:

1) SELECT table_name, column_name FROM gpkg_geometry_columns WHERE table_name IN (SELECT table_name FROM gpkg_contents WHERE data_type == ‘features'))
2) Not testable if result set is empty
3) For each row table_name, column_name from step 1
   h. SELECT sql FROM sqlite_master WHERE type = ‘trigger’ AND tbl_name = ‘fgsi_’ || result_set_table_name || ‘_’ || result_set_column_name
   i. Not testable if result set is empty
   j. /opt/extension_mechanism/extensions/data/table_def
   k. Fail if failed
   l. SELECT extension_name from gpkg_extensions WHERE table_name = result_set_table_name AND column_name = result_set_column_name
   m. Pass if result is “gpkg_srs_id_trigger”
   n. Fail otherwise

Reference: Clause 3.1.5.1.2 Req 80:
Test Type: Basic

A.3.1.5.1.3 SRS_ID Triggers – Extensions Row

Test Case ID: /reg_ext/features/srs_id_triggers/extension_row
Test Purpose: Verify that srs_id triggers are registered using the “gpkg_srs_id_trigger” extension name.
Test Method:

/reg_ext/features/srs_id_triggers/extension_name

Reference: Clause 3.1.5.1.3 Req 81:
Test Type: Capability

A.3.2 Tiles

Field Cod

A.3.2.1 Zoom Levels

A.3.2.1.1 Data

A.3.2.1.1.1 Zoom Other Intervals—Extensions Name

Test Case ID: /reg_ext/tiles/zoom_levels/data/zoom_other_ext_name
Test Purpose: Verify that the "gpkg_zoom_other" extension name is used to register tiles tables with other than powers of two zoom intervals.
Test Method:

1) SELECT table_name FROM geopackage_contents WHERE data_type = 'tiles'
2) Not testable if empty result set
3) For each row table_name from step 1
    a. SELECT zoom_level, pixel_x_size, pixel_y_size FROM tile_matrix_metadata WHERE table_name = selected table name ORDER BY zoom_level ASC
    b. Not testable if returns empty result set
    c. Not testable if there are not two rows with adjacent zoom levels
    d. Not testable if no pair of rows for adjacent zoom levels have pixel_x_size or pixel_y_size values that differ by other than powers of two
    e. /opt/extension_mechanism/extensions/data/table_def
    f. Fail if failed
    g. SELECT * FROM gpkg_extensions WHERE table_name = selected table name AND extension_name = 'gpkg_zoom_other'
    h. Fail if returns an empty result set
    i. Log pass otherwise
4) Pass if logged pass and no fails

Reference: Clause 3.2.1.1.2 Req 82:
Test Type: Basic

A.3.2.1.1.2 Zoom Other Intervals – Extensions Row

Test Case ID: / reg_ext/tiles/zoom_levels/data/zoom_other_ext_row
Test Purpose: Verify that tiles tables with other than powers of two zoom intervals are registered using the "gpkg_zoom_other" extension name.
Test Method:

/reg_ext/tiles/zoom_levels/data/zoom_other_ext_name

Reference: Clause 3.2.1.1.3 Req 83:
Test Type: Capabilty

A.3.2.2 Tile Encoding WEBP

A.3.2.2.1 Data

A.3.2.2.1.1 WEBP – Extensions Name

Test Case ID: /reg_ext/tiles/tile_encoding_webp/data/webp_ext_name

Field Cod

Test Purpose: Verify that the “gpkg_webp” extensions name is used to register WEBP tile encoding implementations.
Test Method:

1) SELECT table_name FROM geopackage_contents WHERE data_type = 'tiles'
2) Not testable if empty result set
3) For each row table_name from step 1
    a. Select tile_data FROM row table_name
    b. For each row tile_data from step a
        i. Log webp if tile data in MIME type image/webp
    c. Not testable if no logged webps
    d. /opt/extension_mechanism/extensions/data/table_def
    e. Fail if failed
    f. SELECT * FROM gpkg_extensions WHERE table_name = selected table name AND extension_name = 'gpkg_webp’
    g. Fail if returns an empty result set
    h. Log pass otherwise
4) Pass if logged pass and no fails

Reference: Clause 3.2.2.2.1 Req 84:
Test Type: Basic

A.3.2.2.1.2 WEBP – Extensions Row

Test Case ID: /reg_ext/tiles/tile_encoding_webp/data/webp_ext_row
Test Purpose: Verify that WEBP tile encodings are registered using the “gpkg_webp” extensions name.
Test Method:

/reg_ext/tiles/tile_encoding_webp/data/webp_ext_name

Reference: Clause 3.2.2.2.2 Req 85:
Test Type: Capability

A.3.2.3 Tile Encoding TIFF

A.3.2.3.1 Data

A.3.2.3.1.1 TIFF – Extensions Name

Test Case ID: /reg_ext/tiles/tile_encoding_tiff/data/tiff_ext_name
Test Purpose: Verify that the “gpkg_tiff” extensions name is used to register TIFF tile encoding implementations.
Test Method:

1) SELECT table_name FROM geopackage_contents WHERE data_type = 'tiles'
2) Not testable if empty result set
3) For each row table_name from step 3
    a. Select tile_data FROM row table_name
    b. For each row tile_data from step a
        i. Log tiff if tile data in MIME type image/tiff
    c. Not testable if no logged webps
    d. /opt/extension_mechanism/extensions/data/table_def

Field Cod

   e. Fail if failed
   f. SELECT * FROM gpkg_extensions WHERE table_name = selected table name AND extension_name = 'gpkg_tiff'
   g. Fail if returns an empty result set
   h. Log pass otherwise
4) Pass if logged pass and no fails

Reference: Clause 3.2.3.1.2 Req 86:
Test Type: Basic

A.3.2.3.1.2 TIFF – Extensions Row

Test Case ID: /reg_ext/tiles/tile_encoding_tiff/data/tiff_ext_row
Test Purpose: Verify that TIFF tile encodings are registered using the "gpkg_tiff" extensions name.
Test Method:

/reg_ext/tiles/tile_encoding_tiff/data/tiff_ext_name

Reference: Clause 3.2.3.1.3 Req 87:
Test Type: Capability

A.3.2.4 Tile Encoding NITF

A.3.2.4.1 Data

A.3.2.4.1.1 NITF – Extensions Name

Test Case ID: /reg_ext/tiles/tile_encoding_nitf/data/nitf_ext_name
Test Purpose: Verify that the "gpkg_nitf" extensions name is used to register NITF tile encoding implementations.
Test Method:

1) SELECT table_name FROM geopackage_contents WHERE data_type = 'tiles'
2) Not testable if empty result set
3) For each row table_name from step 3
   a. Select tile_data FROM row table_name
   b. For each row tile_data from step a
      i. Log nitf if tile data in MIME type application/vnd.NITF
   c. Not testable if no logged webps
   d. /opt/extension_mechanism/extensions/data/table_def
   e. Fail if failed
   f. SELECT * FROM gpkg_extensions WHERE table_name = selected table name AND extension_name = 'gpkg_nitf'
   g. Fail if returns an empty result set
   h. Log pass otherwise
4) Pass if logged pass and no fails

Reference: Clause 3.2.4.1.2 Req 88:
Test Type: Basic

A.3.2.4.1.2 NITF – Extensions Row

Field Cod

Test Case ID: /reg_ext/tiles/tile_encoding/nitf/data/nitf_ext_row
Test Purpose: Verify that NITF tile encodings are registered in the gpkg_extensions table using the "gpkg_nitf" extensions name.
Test Method:

/reg_ext/tiles/tile_encoding_nitf/data/nitf_ext_name

Reference: Clause 3.2.4.1.3 Req 89:
Test Type: Capability

A.3.2.5 Tile Encoding Other

A.3.2.5.1 Data

A.3.2.5.1.1 Other Mime Type – Extensions Name

Test Case ID: /reg_ext/tiles/tile_encoding/other/data/other_ext_name
Test Purpose: Verify that an extension name in the form <author_name>_<other>_mime_type is defined for an author name other than "gpkg" for each other MIME image format used for tile_data columns in tile matrix set user data tables, where <other> is replaced by the other MIME type abbreviation in uppercase.
Test Method:

```
1) SELECT table_name FROM geopackage_contents WHERE data_type = 'tiles'
2) Not testable if empty result set
3) For each row table_name from step 3
      a. Select tile_data FROM row table_name
      b. For each row tile_data from step a
            i. Log other MIME type name if tile data not in MIME type png, jpeg,
               webp, tiff or nitf
      c. Not testable if no logged others
      d. /opt/extension_mechanism/extensions/data/table_def
      e. Fail if failed
      f. For each other logged MIME type name for this table_name
            i. SELECT extension_name FROM gpkg_extensions WHERE
               table_name = result set table name AND column_name = ‘tile_data’
               AND substr(lower(extension_name),1,4) !- ‘gpkg’) AND
               instr(extension_name, logged MIME type name) != 0
           ii. Fail if returns an empty result set
          iii. Separate extension_name into <author> and <extension> at the first "_"
           iv. Separate <extension> into <mime> and <ext> at the first "_"
            v. Fail if <mime> not logged MIME type
           vi. Fail if <ext> not "mime_type"
          vii. Log pass otherwise
4) Pass if logged pass and no fails"
```

Reference: Clause 3.2.5.1.2 Req 90:
Test Type: Basic

A.3.2.5.1.2 Other Mime Type – Extensions Row

Field Cod

Test Case ID: /reg_ext/tiles_tile_encoding/other/data/other_ext_row
Test Purpose: Verify that other mime image type tile encodings are registered in the gpkg_extensions table using names of the form <author_name>_<other>_mime_type.
Test Method:

/reg_ext/tiles/tile_encoding/other/data/other_ext_name

Reference: Clause 3.2.5.1.3 Req 91:
Test Type: Capability

A.3.3 Any Tables

A.3.3.1 Other Trigger

A.3.3.1.1 Data

A.3.3.1.1.1 Other Trigger – Extensions Name

Test Case ID: /reg_ext/any/other_triggers/data/ext_name
Test Purpose: Verify that an extension name in the form <author_name>_<extension> is defined for an author name other than "gpkg" for each other trigger implementation that uses SQL functions other than those provided by SQLite or the GeoPackage Minimal Runtime SQL Functions.
Test Method:

1) SELECT sql, tbl_name FROM sqlite_master WHERE type='trigger'
2) For each row sql, tbl_name from step 1
    a. If sql contains an SQL function other than those provided by SQLite or the GeoPackage Minimal Runtime SQL Functions
        i. Log trigger
        ii. /opt/extension_mechanism/extensions/data/table_def
        iii. Fail if failed
        iv. SELECT extension_name FROM gpkg_extensions WHERE table_name = tbl_name AND substr(lower(extension_name),1,4) !- 'gpkg')
        v. Fail if returns an empty result set
        vi. Log pass otherwise
3) Not testable if no logged triggers
4) Pass if logged pass and no fails

Reference: Clause 3.3.1.1.2 Req 92:
Test Type: Basic

A.3.3.1.1.2 Other Trigger – Extensions Row

Test Case ID: /reg_ext/any/other_triggers/data/ext_row
Test Purpose: Verify that other trigger implementations that use SQL functions other than those provided by SQLite or the GeoPackage Minimal Runtime SQL Functions.are registered with <author_name>_<extension> names in the gpkg_extensions table.
Test Method:

/reg_ext/any/other_triggers/data/ext_name

Reference: Clause 3.3.1.1.3 Req 93:

Field Cod

Test Type: Capability

Field Cod

Annex B Background and Context (Informative)

B.1 Preface

An open standard non-proprietary platform-independent GeoPackage container for distribution and direct use of all kinds of geospatial data will increase the cross-platform interoperability of geospatial applications and web services. Standard APIs for access and management of GeoPackage data will provide consistent query and update results across such applications and services. Increased interoperability and result consistency will enlarge the potential market for such applications and services, particularly in resource-constrained mobile computing environments like cell phones and tablets. GeoPackages will become the standard containers for "MyGeoData" that are used as a transfer format by users and Geospatial Web Services and a storage format on personal and enterprise devices.

This OpenGIS® GeoPackage Implementation Specification defines a GeoPackage as a self-contained, single-file, cross-platform, serverless, transactional, open source SQLite data container with table definitions, relational integrity constraints, an SQL API exposed via a "C" CLI and JDBC, and manifest tables that together act as an exchange and direct-use format for multiple types of geospatial data including vector features, features with raster attributes and tile matrix pyramids, especially on mobile / hand held devices in disconnected or limited network connectivity environments.

Table formats, definitions of geometry types and metadata tables, relational integrity constraints, and SQL API are interdependent specification facets of the SF-SQL [13][11][12] and SQL-MM (Spatial) [14] standards that serve as normative references for the vector feature portion of this specification.

This specification attempts to support and use relevant raster types, storage table definitions, and metadata from widely adopted implementations and existing standards such as WMTS [22] and ISO metadata [42], to integrate use of rasters as attributes of geospatial features, and to define relational integrity constraints and an SQL API thereon to provide a raster analogy to the SF-SQL and SF-MM data access and data quality assurance capabilities.

Conformance classes for this specification are classified as core (mandatory) and extension (optional). The simple core of an Empty GeoPackage contains two SQL tables.

Future versions of this specification may include requirements for elevation data and routes. Future enhancements to this specification, a future GeoPackage Web Service specification, and modifications to existing OGC Web Service (OWS) specifications to use GeoPackages as exchange formats may allow OWS to support provisioning of GeoPackages throughout an enterprise or information community.

B.2 Submitting Organizations

The following organizations submitted this Implementation Specification to the Open Geospatial Consortium Inc. as a Request For Comment (RFC).

a) Envitia
b) Luciad
c) Sigma Bravo
d) The Carbon Project
e) U.S. Army Geospatial Center

Field Cod

f) U.S. National Geospatial Intelligence Agency

B.3 Revision history

Date	**Rel**	**Editor**	**Paragraph modified**	**Description**
2012-11-15	r1	Paul Daisey	10.3	Remove min/max x/y fields from all tables and text in clause 10.3 Tile Table Metadata per change request 250 / 12-135.
2012-11-15	r1	Paul Daisey	10.2, Annex B	add compr_qual_factor and georectification columns to raster_columns table create statement and sample insert statement; add triggers for those columns matching those for _rt_metadata per change request 251 / 12-134
2013-01-15	r2	Paul Daisey	8.2	Change gpkg_contents description default value per change request 255 / 12-166
2013-01-15	r2	Paul Daisey	9.2, Annex A	SRS Table Name Change per change request 256 / 12-165
2013-01-16	r2	Paul Daisey	7, Figure 2	table diagram gpkg_contents min_y REAL instead of BLOB
2013-01-23	r2	Paul Daisey	11.3, 8.2	Clause reference corrections, change gpkg_contents.identifier default value to “”
2013-02-01	r2	Paul Daisey	Changes to AS	No changes to AS
2013-02-01	r2	Paul Daisey	8.2	new last sentence and NOTE1, additional table name triggers
2013-02-01	r2	Paul Daisey	9.6	drop tables 21, 22 and associated text
2013-02-01	r2	Paul Daisey	10.5	misc. editorial changes
2013-02-01	r2	Paul Daisey	11.2	REQ 71 should refer to clause 11.2 and not 11.1
2013-02-01	r2	Paul Daisey	12	new clause 12 other data
2013-02-01	r2	Paul Daisey	13.2	rename tables 56,57 remove “ows_” prefix
2013-02-08	r3	Paul Daisey	10.2, 10.7, 10.8	Use -1 as "magic" value indicating "unknown" for both compr_qual_factor and georectification columns, and make it the default value. Remove NOTE1 in 10.7. Delete the next to last row in Table 46 - Image Routines for gpkgBboxToTiles ().

Field Cod

				Delete the corresponding sub-clause 10.8.10 Renumber sub-clause 10.8.11 to 10.8.10
2013-02-22	R3	Paul Daisey	Normative References, Future Work, 6, Bibliography	The GeoPackage file format and SQL API are provided by SQLite, which is the GeoPackage container implementation, not just a a reference implementation.
2013-03-05	R3	Paul Daisey	6.4	Add Security Considerations clause.
2013-03-05	R3	Paul Daisey	Future Work	Streaming synchronization
2013-03-30	R3	Paul Daisey	Normative References, All, Bibliography	Move references to geos and proj4 libraries from Normative References to Bibliography, remove references to them from main text.
2013-03-30 2013-04-01	R3	Paul Daisey	Reorganize document and Annexes	New Core / Extension outline.
2013-03-30	R3	Paul Daisey	6.3.2.2	auth_name column case-insensitive
2013-03-30	R3	Paul Daisey		Add feature table layout example
2013-04-01	R3	Paul Daisey	All,Annex B	Move table definition SQL to Annex B
2013-04-01	R3	Paul Daisey	7.2.4	Remove requirements for SQL/MM functions, REQ 21 – 33.
2013-04-03	R3	Paul Daisey	All	Renumber tables, figures, normative references
2013-04-09	R4	Paul Daisey	6.3.6	Make integer primary keys mandatory for user vector, raster and tile data tables.
2013-04-09	R4	Paul Daisey	6.3.3.2,	Rewrite clause, remove references to geometry_columns table columns that are superfuluous in SQLite implementation.
2013-04-09	R4	Paul Daisey	6.3.6.1	Rewrite clause, remove references to SF/SQL gS and gB architectures.
2013-04-18	R4	Paul Daisey	6.3.4.1, 6.3.4.2, 6.3.6.3	Remove normative references to RasterLite
2013-04-19	R4	Paul Daisey	6.3.6.3	GeoPackage description of other data tables.
2013-04-29	R4	Paul Daisey	All	Remove implementation references
2013-04-29	R4	Paul Daisey	6.3.6.3, Annex G	Remove manifest other data entries
2013-04-29	R4	Paul Daisey	6.3.2.4.2, Annex B, E	Allow metadata of specified MIME type
2013-04-29	R4	Paul Daisey	6.3.2.4.3, Annex B, E	Allow NULLs in metadata_reference table

Field Cod

2013-04-29	R4	Paul Daisey	6.3.3.2, new Annex F	Geometry type codes
2013-04-29	R4	Paul Daisey	6.3.2.4, new Annex L	Feature Schema Metadata example
2013-05-03	R5	Paul Daisey	Future Work	Geographic / Geodetic Geometries
201305-07	R5	Paul Daisey	6.3.4.2, Annex C, E	Remove compr_qual_factor and georectification columns from raster_columns table
2013-05-07	R5	Paul Daisey	6.3.2.4, 6.3.4.3, new Annex M	delete _rt_metadata tables add Annex M reference Annex M from note in 6.3.2.4
2013-05-07	R5	Paul Daisey	7.1.1, Normative References, Bibliography	Add NITF as an extension image format
2013-05-07	R5	Paul Daisey	6.3.1	Revise Table Diagram
2013-05-07	R5	Paul Daisey	7.3.3, Annex C	Remove raster functions
2013-05-11	R5	Paul Daisey	6.3.2.4.3	metadata_reference table is not required to contain any rows
2013-05-11	R5	Paul Daisey	6.3.2.4.2	Recommend ISO 19139 metadata
2013-05-11	R5	Paul Daisey	6.3, Annex B	Default values
2013-05-11	R5	Paul Daisey	7.3.3, Annex C	Minimal Runtime SQL Functions
2013-05-11	R5	Paul Daisey	7.3.4, Annex D	Spatial Indexes
2013-05-13	R5	Paul Daisey	6, 7	Reformat requirement tables, unduplicate requirement text
2013-05-15	R5	Paul Daisey	6.3.1, 6.3.2.4, 6.3.4.2, 7.3.5.5, Annex B, remove Annex L	Replace raster_columns table, Annex L with gpkg_data_columns table
2013-05-16	R5	Paul Daisey	6.3.2.3, 7.4, Annex G,H,I	Drop manifest table, schemas, sample document
2013-05-16	R5	Paul Daisey	Future Work	Add GeoPackage Abstract Object Model
2013-05-22	R5	Paul Daisey	6.2.1, 6.3.3.1, new 7.1.1, Annex F	Add optional support for non-linear geometry types
2013-05-22	R5	Paul	7.3.2	Add SQLite configuration requirements

Field Cod

		Daisey		
2013-05-22	R5	Paul Daisey	6.3, 7.2	Require only gpkg_contents and spatial_ref_sys tables
2013-05-24	R5	Paul Daisey	7.2.1.3	Add gpkg_extensions table
2013-05-24	R5	Paul Daisey	7.3.4, Annex D	Provide spatial index templates instead of examples
2013-05-25	R5	Paul Daisey	Preface, Scope, Terms, 6, 7	Simplify, rewrite, add terms, use terms
2013-05-26	R5	Paul Daisey	All	Incorporate terms, renumber requirements and tables
2013-05-28	R5	Paul Daisey	6.1.2	Add "GPKG" as SQLite application id
2013-05-28	R5	Paul Daisey	6.1.2	Add SQLITE PRAGMA integrity_check
2013-05-28	R5	Paul Daisey	6.2.1	Geometry format minor changes
2013-05-28	R5	Paul Daisey	6.3.2.2, Annex E	Remove references to raster_columns table (removed previously)
2013-05-28	R5	Paul Daisey	All	Clause number references and text changes required by 5/22 changes
2013-05-28	R5	Paul Daisey	All	Remove comments on accepted changes
2013-05-28	R5	Paul Daisey	Annex E E.4	Add non-linear geometry type codes
2013-05-29	R5	Paul Daisey	7.2.4.1	Change reference from SF/SQL to SQL/MM
2013-05-29	R5	Paul Daisey	All	Change core and extension requirement names required by 5/22 changes
2013-05-29	R5	Paul Daisey	Table 16	Change extension to API to avoid overloading extension term
2013-05-29	R5	Paul Daisey	A.2	Draft changes to A.2 Conformance Classes
2013-05-29	R5	Paul Daisey	B.3	Add gpkg_data_columns table SQL
2013-05-30	R5	Paul Daisey	Revision History	Record 5/29 changes
2013-06-06	R6	Paul Daisey	Preface, Submission Contact Points, Revision History, Changes to AS, Changes to IS, Future Work, Forward,	Remove all forward material except title page, submitting orgs, and introduction, and put in annexes.

Field Cod

			Introduction, Clauses 1-5	
2013-06-07	R6	Paul Daisey	Old Clauses 6,7 -> New 1-3	Restructure document iaw draft Requirements Dependencies
2013-06-07	R6	Paul Daisey	Annex A	Revised Requirements Dependencies and Diagram
2013-06-10	R6	Paul Daisey	All	Fix clause and requirement references based on document restructure
2013-06-10	R6	Paul Daisey	Annex A	Add Abstract Test Suite (incomplete)
2013-06-11	R6	Paul Daisey	Clause 1,2, Annex A	Insert Base and Extension subclauses, renumber more deeply nested subclauses
2013-06-12	R6	Paul Daisey	Annex G	Remove names and codes for Z and M geometry types, add Figure 5 and geometry subtype definitions
2013-06-12	R6	Paul Daisey	Clause 1.2.2.6	Rewrite clause, add new Requirement 10, 11, renumber existing and subsequent ones.
2013-06-12	R6	Paul Daisey	Annex D	Add ST_Is3D() and ST_IsMeasured()
2013-06-12	R6	Paul Daisey	All	Add “gpkg_” prefix to all GeoPackage metadata tables
2013-06-12	R6	Paul Daisey	Figure 1, 2	Update with “gpkg_” prefix
2013-06-12	R6	Paul Daisey	Annex A	Add Abstract Test Suite (incomplete)
2013-06-13	R6	Paul Daisey	1.2.4.1	Add sentence to end of first paragraph describing gpkg_other_data_columns content..
2013-06-13	R6	Paul Daisey	Annex A	Add Abstract Test Suite (incomplete)
2013-06-17	R6	Paul Daisey	Clause 1,2,3	Revised notes and turned them into footnotes; moved normative text into requirement statements.
2013-06-20	R6	Paul Daisey	All	Restructure document iaw SpecificationStructureAlternative3
2013-06-24	R6	Pepijn Van Eeckhoudt	All	Created and applied Word Styles and Outline List Numbering
2013-06-26	R6	Paul Daisey	1.1.2, 2.1.1, 2.1.4, 3.1.2, Annex C, D, F, G	GeoPackage Geometry Encoding Revisions
2013-06-27	R6	Paul Daisey	3.1.3.1.1	Add footnote recommendation on Spatial Index drop/add if many updates.
2013-06-27	R6	Paul Daisey	Figure 1, 2.2.6, 2.2.7	Remove gpkg_tile_table_metadata table
2013-06-28	R6	Paul	All	Change requirement statement format to

Field Cod

		Daisey		**Req # s *SHALL o in bold italic***
2013-06-28	R6	Paul Daisey	Annex B	Update definition of Empty GeoPackage, add definition of Valid GeoPackage
2013-06-28	R6	Paul Daisey	Figure 1, 2.2.7, Annex C, F	Change tile_matrxI_metadata t_table_name column name to table_name iaw changes to gpkg_geometry_columns column name changes.
2013-06-28	R6	Paul Daisey	Figure 1, 2.1.5, 2.2.7, Annex C, F	Add gpkg_geometry_columns and gpkg_tile_matrix_metadata table_name foreign key constraints referencing gpkg_contents table_name now that gpkg_contents rows may describe other data tables.
2013-06-28	R6	Paul Daisey	Clause 3	Tables with non “gpkg” author registered extensions not data_type “features” or “tiles”
2013-07-01	R7	Paul Daisey	Annex A	Change ATS format from numbered list to bold heading, add test definitions.
2013-07-02	R7	Paul Daisey	Annex A	Add test definitions.
2013-07-03	R7	Paul Daisey	Annex A	Revise, add test definitions.
2013-07-04	R7	Paul Daisey	1.1.1, Annex A	Change .geopackage to .gpkg
2013-07-24	R7	Paul Daisey	Annex B	Add “Potential” to “Future Work”, “MAY” to items.
2013-07-24	R7	Paul Daisey	Annex B	Add support for UTFGrid as a future work item.
2013-07-24	R7	Paul Daisey	1.1.1.1.1	Add footnote to REQ 1 that SQLite is in the public domain.
2013-07-24	R7	Paul Daisey	2.1.3.1.1	Add footnote to Table 4 that OGC WKB is subset of ISO WKB
2013-07-24	R7	Paul Daisey	2.1.3.1.1	Revise definition of geometry type in Table 4 to include is_empty flag; add paragraph on encoding empty point geometries.
2013-07-24	R7	Paul Daisey	Annex E	Revise spatial index triggers to handle NULL values.
2013-07-31	R7	Paul Daisey	Annex C, F	Correct SQL errors in tables 13, 32, 43
2013-07-31	R7	Paul Daisey	Annex D	Add ST_IsEmpty(geom. Geometry)
2013-07-31	R7	Paul Daisey	Annex E Table 39	Revise spatial index triggers to handle empty geometries, changed ROWID values.
2013-07-31	R7	Paul Daisey	Annex A A.3.1.3.1.1	Revise test method iaw changes to spatial index triggers
2013-07-31	R7	Paul Daisey	2.1.3.1.1	Envelope in geopackage geometry binary for empty geometry

Field Cod

2013-07-31	R7	Paul Daisey	Annex A A.2.1.2.1.1	Revise test method to test for NaN values in envelope of empty geometries
2013-08-01	R8	Paul Daisey	Submitting Organizations, Submission Contact Points	Moved Submitting Organizations to B2; deleted previous B2 Submission Contact Points
2013-08-01	R8	Paul Daisey	1.1.3.1.1 Table 3, 2.1.6.1.2, Annex A, C	Nullable gpkg_contents columns One geometry column per feature table.

B.4 Changes to the OGC® Abstract Specification

The OGC® Abstract Specification does not require changes to accommodate this OGC® standard.

B.5 Changes to OpenGIS® Implementation Standards

None at present.

B.6 Potential Future Work

- MAY investigate GeoPackage implementation on SQLite version 4 [B46].
- Future versions of this specification MAY include requirements for elevation data and routes.
- Future enhancements to this specification, a future GeoPackage Web Service specification and modifications to existing OGC Web Service (OWS) specifications to use GeoPackages as exchange formats MAY allow OWS to support provisioning of GeoPackages throughout an enterprise.
- Future versions of this specification MAY include additional raster / image formats, including fewer restrictions on the image/tiff format.
- Future versions of this specification MAY include additional SQL API routines for interrogation and conversion of raster / image BLOBs.
- Future versions of this specification and/or one for a GeoPackage Web Service MAY address utilities for importing and exporting vector, raster and tile data in various formats.
- Future versions of this specification and/or one for a GeoPackage Web Service MAY address encryption of GeoPackages and/or individual tables or column values.
- Future versions of this specification MAY add infrastructure to the metadata tables such as a temporal_columns table that refers to the time properties of data records.
- MAY specify a streaming synchronization protocol for GeoPackage as part of a future GeoPackage Web Service specification, and/or a future version of the GeoPackage and/or Web Synchronization Service specification(s).

Field Cod

- Future versions of this specification MAY address symbology and styling information.
- Future version of this specification MAY include geographic / geodesic geometry types.
- MAY create a GeoPackage Abstract Object Model to support data encodings other than SQL in a future version of this specification.
- MAY add UTFGrid (https://github.com/mapbox/utfgrid-spec) support in a future version of this specification

B.7 Contributors

The following organizations and individuals have contributed to the preparation of this standard:

- Alessandro Furieri
- Compusult Limited
- Development Seed
- Environmental Systems Research Institute, Inc. (Esri)
- Envitia
- Feng China University
- George Mason University
- Image Matters LLC
- International Geospatial Services Institute (iGSI) GmbH
- LMN Solutions
- Luciad
- MapBox
- OpenGeo
- Open Site Plan
- Sigma Bravo
- The Carbon Project
- Universitat Autònoma de Barcelona (CREAF)
- U.S. Army Geospatial Center (AGC)
- U.S. National Aeronautics and Space Administration (NASA)
- U.S. National Geospatial Intelligence Agency (NGA)

B.8 Terms and Definitions

For the purposes of this document, the following terms and definitions apply.

B.8.1 Empty GeoPackage

A GeoPackage file that contains a spatial_ref_sys table, agpkg_contents table with row record(s) with data_type column values of “features” or “tiles”, and corresponding features tables per clause 2.1 and/or tiles tables per clause 2.2 where the user data tables per clauses 2.1.6 and 2.2.7 exist but contain no rows.

B.8.2 geolocate

identify a real-world geographic location

B.8.3 GeoPackage

Field Cod

A GeoPackage file used with a GeoPackage SQLite Extension

B.8.4 GeoPackage file

a platform-independent SQLite database file that contains GeoPackage data and metadata tables with specified definitions, integrity assertions, format limitations and content constraints.

B.8.5 GeoPackage SQLite Extension

executable software linked to a SQLite library with specified configuration requirements to provide SQL API access to and functional manipulation of GeoPackage file contents.

B.8.6 georectified

raster whose pixels have been regularly spaced in a geographic (i.e., latitude / longitude) or projected map coordinate system using ground control points so that any pixel can be geolocated given its grid coordinate and the grid origin, cell spacing, and orientation.

B.8.7 orthorectified

georectified raster that has also been corrected to remove image perspective (camera angle tilt), camera and lens induced distortions, and terrain induced distortions using camera calibration parameters and DEM elevation data to accurately align with real world coordinates, have constant scale, and support direct measurement of distances, angles, and areas.

B.8.8 Valid GeoPackage

A GeoPackage file that contains features per clause 2.1 and/or tiles per clause 2.2 and row(s) in the gpkg_contents table with data_type column values of “features” and/or “tiles” describing the user data tables.

B.9 Conventions

Symbols (and abbreviated terms)

Some frequently used abbreviated terms:

ACID	Atomic, Consistent, Isolated, and Durable
ASCII	American Standard Code for Information Interchange
API	Application Program Interface
ATOM	Atom Syndication Format
BLOB	Binary Large OBject
CLI	Call-Level Interface
COTS	Commercial Off The Shelf
DEM	Digital Elevation Model
DIGEST	Digital Geographic Information Exchange Standard

Field Cod

GeoTIFF	Geographic Tagged Image File Format
GPKG	GeoPackage
GRD	Ground Resolved Distance
EPSG	European Petroleum Survey Group
FK	Foreign Key
IETF	Internet Engineering Task Force
IIRS	Image Interpretability Rating Scale
IRARS	Imagery Resolution Assessments and Reporting Standards (Committee)
ISO	International Organization for Standardization
JDBC	Java Data Base Connectivity
JPEG	Joint Photographics Expert Group (image format)
MIME	Multipurpose Internet Mail Extensions
NATO	North Atlantic Treaty Organization
NITF	National Imagery Transmission Format
OGC	Open Geospatial Consortium
PK	Primary Key
PNG	Portable Network Graphics (image format)
RDBMS	Relational Data Base Management System
RFC	Request For Comments
SQL	Structured Query Language
SRID	Spatial Reference (System) Identifier
TIFF	Tagged Image File Format
TIN	Triangulated Irregular Network
UML	Unified Modeling Language
UTC	Coordinated Universal Time
XML	eXtensible Markup Language
1D	One Dimensional
2D	Two Dimensional
3D	Three Dimensional

B.10 UML Notation

The diagrams that appear in this standard are presented using the Unified Modeling Language (UML) static structure diagrams. The UML notations used in this standard for RDBMS tables in the GeoPackage container are described in Figure 1 below.

Field Cod

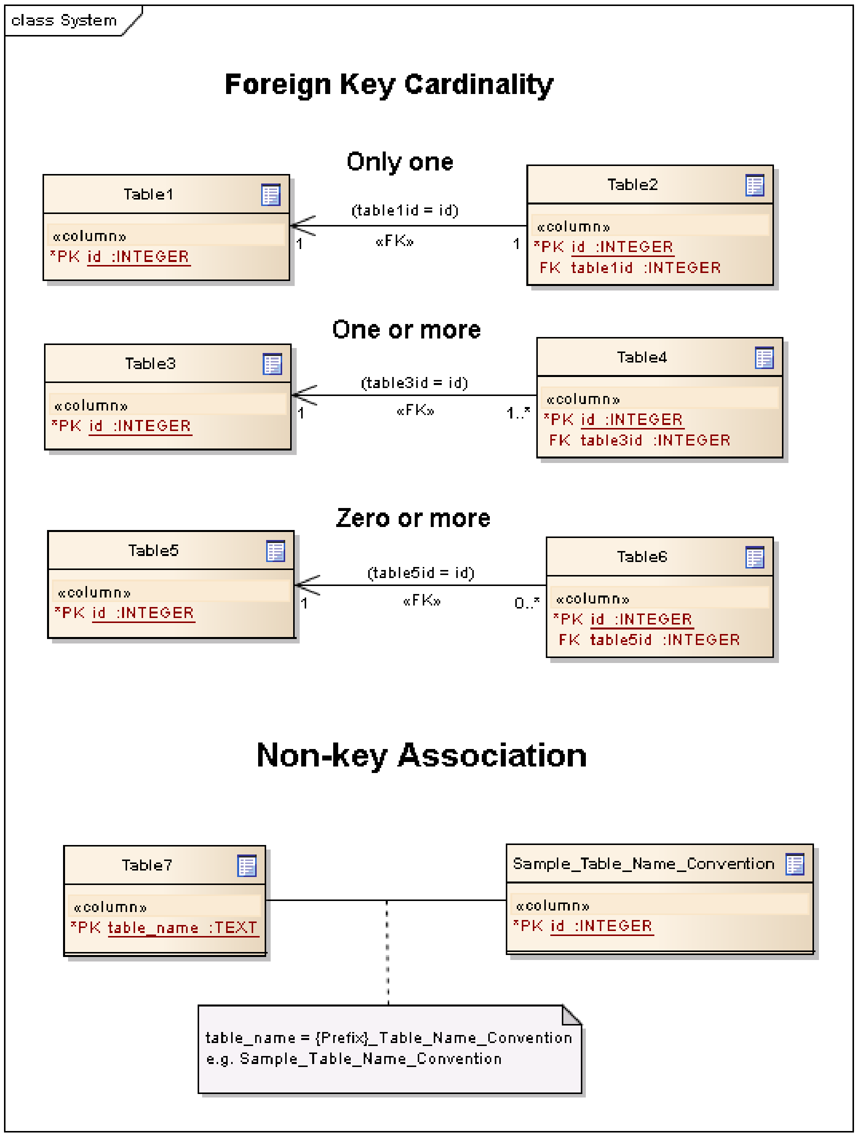


Figure 3 - UML Notation for RDBMS Tables

In this standard, the following two stereotypes of UML classes are used to represent RDBMS tables:

a) <<table>> An instantiation of a UML class as an RDMBS table.

b) <<column>> An instantiation of a UML attribute as an RDBMS table column.

In this standard, the following standard data types are used for RDBMS columns:

a) NULL – The value is a NULL value.

b) INTEGER – A signed integer, stored in 1, 2, 3, 4, 6, or 8 bytes depending on the magnitude of the value

Field Cod

c) REAL – The value is a floating point value, stored as an 8-byte IEEE floating point number.

d) TEXT – A sequence of characters, stored using the database encoding

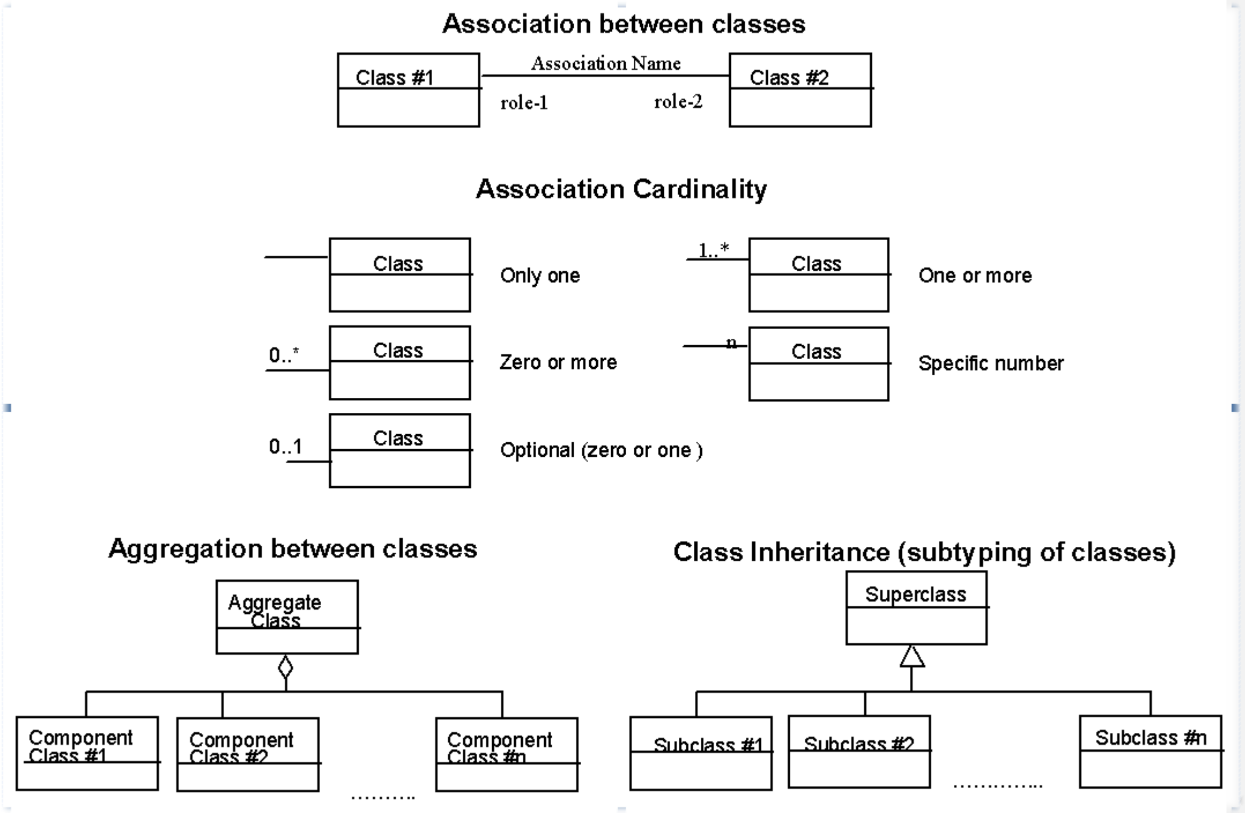
(UTF-8, UTF-16BE or UTF-16LE).

e) BLOB – The value is a blob of data, stored exactly as it was input.

f) NONE – The value is a Date / Time Timestamp

The UML notations used in this standard for the eXtensible Markup Language (XML) schema for the GeoPackage manifest are described in Figure 2 below.

Figure 3 UML notation for XML Schema



In this standard, the following stereotypes of UML classes are used to describe XML schemas:

a) <<DataType>> A descriptor of a set of values that lack identity (independent existence and the possibility of side effects). A DataType is a class with no operations whose primary purpose is to hold the information.

b) <<Enumeration>> is a fixed enumeration that uses string values for expressing a list of potential values.

Field Cod

c) <<CodeList>> is an extensible enumeration that uses string values for expressing a list of potential values.

In this standard, the standard data types defined in XML Schema Part 2: Datatypes[41]are used to describe XML Schemas.

Field Cod

Annex C Table Definition SQL (Normative)

C.1 gpkg_spatial_ref_sys

Table 19 gpkg_spatial_ref_sys Table Definition SQL

```
CREATE TABLE gpkg_spatial_ref_sys (
  Srs_name TEXT NOT NULL,
  Srs_id INTEGER NOT NULL PRIMARY KEY,
  organization TEXT NOT NULL,
  organization_coordsys_idINTEGER NOT NULL,
  definition  TEXT NOT NULL,
  description TEXT
)
```

Table 20 SQL/MM View of gpkg_spatial_ref_sys Definition SQL

```
CREATE VIEW st_spatial_ref_sys AS SELECT
  srs_name,
  srs_id,
  organization,
  organization_coordsys_id,
  definition,
  description
FROM gpkg_spatial_ref_sys;
```

Table 21 SF/SQL View of gpkg_spatial_ref_sys Definition SQL

```
CREATE VIEW spatial_ref_sys AS SELECT
  srs_id AS srid,
  organization AS auth_name,
  organization_coordsys_id AS auth_srid,
  definition AS srtext
  FROM gpkg_spatial_ref_sys;
```

C.2 gpkg_contents

Table 22 gpkg_contents Table Definition SQL

```
CREATE TABLE gpkg_contents (
  table_name TEXT NOT NULL PRIMARY KEY,
  data_type TEXT NOT NULL,
  identifier TEXT UNIQUE,
  description TEXT DEFAULT '',
  last_change TEXT NOT NULL DEFAULT
    (strftime('%Y-%m-%dT%H:%M:%fZ',CURRENT_TIMESTAMP)),
```

Field Cod

```
  min_x DOUBLE,
  min_y DOUBLE,
  max_x DOUBLE,
  max_y DOUBLE,
  srs_id INTEGER,
  CONSTRAINT fk_gc_r_srs_id FOREIGN KEY (srs_id) REFERENCES
    gpkg_spatial_ref_sys(srs_id))
```

C.3 gpkg_extension

Table 23 gpkg_extensions Table Definition SQL

```
CREATE TABLE gpkg_extensions (
  table_name TEXT,
  column_name TEXT,
  extension_name TEXT NOT NULL,
  CONSTRAINT ge_tce UNIQUE
    (table_name, column_name, extension_name))
```

C.4 gpkg_geometry_columns

Table 24 gpkg_geometry_columns Table Definition SQL

```
CREATE TABLE gpkg_geometry_columns (
  table_name TEXT NOT NULL,
  column_name TEXT NOT NULL,
  geometry_type TEXT NOT NULL,
  srs_id INTEGER NOT NULL,
  z INTEGER NOT NULL,
  m INTEGER NOT NULL,
  CONSTRAINT pk_geom_cols PRIMARY KEY (table_name, column_name),
  CONSTRAINT fk_gc_tn FOREIGN KEY (table_name)
                       REFERENCES gpkg_contents(table_name),
  CONSTRAINT fk_gc_srs FOREIGN KEY (srs_id)
                       REFERENCES gpkg_spatial_ref_sys (srs_id))
```

Table 25 SQL/MM View of gpkg_geometry_columns Definition SQL

```
CREATE VIEW st_geometry_columns AS SELECT
  table_name,
  column_name,
  "ST_" || geometry_type_name,
  g.srs_id,
  srs_name
FROM gpkg_geometry_columns as g JOIN gpkg_spatial_ref_sys AS s
WHERE g.srs_id = s.srs_id;
```

Field Cod

Table 26 SF/SQL VIEW of gpkg_geometry_columns Definition SQL

```
CREATE VIEW geometry_columns AS SELECT
  table_name AS f_table_name,
  column_name AS f_geometry_column,
  code4name[1](geometry_type_name) AS geometry_type,
  2 + (CASE z WHEN 1 THEN 1 WHEN 2 THEN 1 ELSE 0 END)
    + (CASE m WHEN 1 THEN 1 WHEN 2 THEN 1 ELSE 0 END)
    AS coord_dimension,
  srs_id AS srid
FROM gpkg_geometry_columns;
```

C.5 sample_feature_table (Informative)

Table 27 sample_feature_table Table Definition SQL

```
CREATE TABLE sample_feature_table (
  id INTEGER AUTOINCREMENT PRIMARY KEY,
  geometry_one BLOB NOT NULL,
  text_attribute TEXT NOT NULL,
  real_attribute REAL NOT NULL,
  numeric_attribute NUMERIC NOT NULL,
  raster_or_photo BLOB NOT NULL
)
```

C.6 gpkg_tile_matrix_metadata

Table 28 gpkg_tile_matrix_metadata Table Creation SQL

```
CREATE TABLE gpkg_tile_matrix_metadata (
  table_name TEXT NOT NULL,
  zoom_level INTEGER NOT NULL,
  matrix_width INTEGER NOT NULL,
  matrix_height INTEGER NOT NULL,
  tile_width INTEGER NOT NULL,
  tile_height INTEGER NOT NULL,
  pixel_x_size DOUBLE NOT NULL,
pixel_y_size DOUBLE NOT NULL,
CONSTRAINT pk_ttm PRIMARY KEY (table_name, zoom_level)
  ON CONFLICT ROLLBACK,
CONSTRAINT fk_tmm_table_name FOREIGN KEY (table_name)
  REFERENCES gpkg_contents(table_name))
```

Table 29 EXAMPLE: gpkg_tile_matrix_metadata Insert Statement

```
INSERT INTO gpkg_tile_matrix_metadata VALUES (
"sample_matrix_tiles",
```

[1] Implementer must provide code4name(geometry_type_name) SQL function

Field Cod

```
0,
1,
1,
512,
512,
2.0,
2.0);
```

C.7 sample_matrix_tiles (Informative)

Table 30 EXAMPLE: tiles table Create Table SQL

```
CREATE TABLE sample_matrix_tiles (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  zoom_level INTEGER NOT NULL,
  tile_column INTEGER NOT NULL,
  tile_row INTEGER NOT NULL,
  tile_data BLOB NOT NULL),
UNIQUE (zoom_level, tile_column, tile_row))
```

Table 31 EXAMPLE: tiles table Insert Statement

```
INSERT INTO sample_matrix_tiles VALUES (
1,
1,
1,
1,
"BLOB VALUE")
```

C.8 gpkg_data_columns

Table 32 gpkg_data_columns Table Definition SQL

```
CREATE TABLE gpkg_data_columns (
  table_name TEXT NOT NULL,
  column_name TEXT NOT NULL,
  name TEXT,
  title TEXT,
  description TEXT,
  mime_type TEXT,
  CONSTRAINT pk_gdc PRIMARY KEY (table_name, column_name)
  ON CONFLICT ROLLBACK,
  CONSTRAINT fk_gdc_tn FOREIGN KEY (table_name)
           REFERENCES gpkg_contents(table_name))
```

C.9 metadata

Table 33 gpkg_metadata Table Definition SQL

```
CREATE TABLE gpkg_metadata (
  id INTEGER CONSTRAINT m_pk PRIMARY KEY ASC
    ON CONFLICT ROLLBACK AUTOINCREMENT NOT NULL UNIQUE,
```

Field Cod

```
  md_scope TEXT NOT NULL DEFAULT 'dataset',
  metadata_standard_URI TEXT NOT NULL DEFAULT
'http://schemas.opengis.net/iso/19139/',
  mime_type TEXT NOT NULL DEFAULT ‘text/xml’,
  metadata TEXT NOT NULL DEFAULT (‘’)
)
```

C.10 metadata_reference

Table 34 gpkg_metadata_reference Table Definition SQL

```
CREATE TABLE gpkg_metadata_reference (
  reference_scope TEXT NOT NULL,
  table_name TEXT,
  column_name TEXT,
  row_id_value INTEGER,
  timestamp TEXT NOT NULL DEFAULT (strftime('%Y-%m-
%dT%H:%M:%fZ',CURRENT_TIMESTAMP)),
  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)
)
```

Table 35 Example: gpkg_metadata_reference SQL insert statement

```
INSERT INTO gpkg_metadata_reference VALUES (
'table','sample_rasters',NULL, NULL, '2012-08-
17T14:49:32.932Z', 98, 99)
```

Field Cod

Annex D Minimal Runtime SQL Functions (Normative)

Table 36 Minimal Runtime SQL Functions

SQL Function	Description	Use
ST_SRID(geom. Geometry) : INTEGER	Returns the spatial reference system id of a Geometry	Check that geometry srid matches what's specified in geometry_columns.srid
ST_GeometryType(geom. Geometry) : TEXT	Returns the WKB geometry type name of a Geometry	Check that the geometry type matches what's specified in geometry_columns.geometry_type
GPKG_IsAssignable(expected_type_name TEXT, actual_type_name TEXT): INTEGER	Returns 1 if a value of type expected_type_name is the same or a super type of type actual_type_name. Returns 0 otherwise.	Determine if the expected geometry type is the same as or a super type of the actual geometry type.
ST_IsEmpty(geom. Geometry)	Returns 1 if geometry value is empty, 0 if not empty, NULL if geometry value is NULL.	Test if a geometry value corresponds to the empty set
ST_Is3D(geom. Geometry)	Tests whether a Geometry value has z coordinates	Check that elevation values are present when they are required, and not present when they are prohibited.
ST_IsMeasured(geom. Geometry)	Tests whether a Geometry value has m coordinates.	Check that measure values are present when they are required, and not present when they are prohibited.
ST_MinX(geom. Geometry) : REAL	Returns the minimum X value of the bounding Envelope of a Geometry	Update the spatial index on a geometry column in a feature table.
ST_MaxX(geom. Geometry) : REAL	Returns the maximum X value of the bounding Envelope of a Geometry	Update the spatial index on a geometry column in a feature table.
ST_MinY(geom. Geometry) : REAL	Returns the minimum Y value of the bounding	Update the spatial index on a geometry column in a feature table.

Field Cod

	Envelope of a Geometry	
ST_MaxY(geom. Geometry) : REAL	Returns the maximum Y value of the bounding Envelope of a Geometry	Update the spatial index on a geometry column in a feature table.
ST_MinZ(geom. Geometry) : REAL	Returns the minimum Z value of the bounding Envelope of a Geometry	Update the spatial index on a geometry column in a feature table.
ST_MaxZ(geom. Geometry) : REAL	Returns the maximum Z value of the bounding Envelope of a Geometry	Update the spatial index on a geometry column in a feature table.
ST_MinM(geom. Geometry) : REAL	Returns the minimum M value of the bounding Envelope of a Geometry	Update the spatial index on a geometry column in a feature table.
ST_MaxM(geom. Geometry) : REAL	Returns the maximum M value of the bounding Envelope of a Geometry	Update the spatial index on a geometry column in a feature table.

Field Cod

Annex E Spatial Index SQL (Normative)

E.1 Create Virtual Table

GeoPackage files MAY contain a spatial index for each geometry column of a feature table. These spatial indexes SHALL be created using the SQLite Virtual Table RTree extension. An application that creates a spatial index SHALL create it using the following SQL statement template:

Table 37 create virtual table SQL

```
CREATE VIRTUAL TABLE rtree_<t>_<c>
 USING rtree(id, minx, maxx, miny, maxy)
```

where <t> and <c> are replaced with the names of the feature table and geometry column being indexed. The rtree function id parameter becomes the virtual table 64-bit signed integer primary key id column, and the min/max x/y parameters are min- and max-value pairs (stored as 32-bit floating point numbers) for each dimension that become the virtual table data columns that are populated to create the spatial rtree index.

E.2 Load Spatial Index Values

The indexes provided by the SQLite Virtual Table RTree extension are not automatic indices. This means the index data structure needs to be manually populated, updated and queried.

Each newly created spatial index SHALL be populated using the following SQL statement

Table 38 load spatial index values SQL

```
INSERT OR REPLACE INTO rtree_<t>_<c>
SELECT rowid, st_minx(<c>), st_maxx(<c>),
              st_miny(<c>), st_maxy(<c>)
FROM <t>;
```

where <t> and <c> are replaced with the names of the feature table and geometry column being indexed.

E.3 Define Triggers to Maintain Spatial Index Values

For each spatial index in a GeoPackage file, corresponding insert, update and delete triggers that update the spatial index SHALL be present on the indexed geometry column. These spatial index triggers SHALL be defined as follows:

Table 39 Spatial Index SQL Triggers Template

```
/* Conditions: Insertion of non-empty geometry
   Actions   : Insert record into rtree */
CREATE TRIGGER rtree_<t>_<c>_insert AFTER INSERT ON <t>
  WHEN (new.<c> NOT NULL AND NOT ST_IsEmpty(NEW.<c>))

BEGIN
  INSERT OR REPLACE INTO rtree_<t>_<c> VALUES (
    NEW.rowid,
    ST_MinX(NEW.<c>), ST_MaxX(NEW.<c>),
```

Field Cod

```
    ST_MinY(NEW.<c>), ST_MaxY(NEW.<c>)
  );
END;

/* Conditions: Update of geometry column to non-empty geometry
               No row ID change
   Actions   : Update record in rtree */
CREATE TRIGGER rtree_<t>_<c>_update1 AFTER UPDATE OF <c> ON <t>
  WHEN OLD.rowid = NEW.rowid AND
      (NEW.<c> NOTNULL AND NOT ST_IsEmpty(NEW.<c>))
BEGIN
    INSERT OR REPLACE INTO rtree_<t>_<c> VALUES (
      NEW.rowid,
      ST_MinX(NEW.<c>), ST_MaxX(NEW.<c>),
      ST_MinY(NEW.<c>), ST_MaxY(NEW.<c>)
    );
 END;

/* Conditions: Update of geometry column to empty geometry
               No row ID change
   Actions   : Remove record from rtree */
CREATE TRIGGER rtree_<t>_<c>_update2 AFTER UPDATE OF <c> ON <t>
  WHEN OLD.rowid = NEW.rowid AND
      (NEW.<c> ISNULL OR ST_IsEmpty(NEW.<c>))
BEGIN
   DELETE FROM rtree_<t>_<c> WHERE id = OLD.rowid;
 END;

/* Conditions: Update of any column
               Row ID change
               Non-empty geometry
   Actions   : Remove record from rtree for old rowid
               Insert record into rtree for new rowid */
CREATE TRIGGER rtree_<t>_<c>_update3 AFTER UPDATE OF <c> ON <t>
  WHEN OLD.rowid != NEW.rowid AND
      (NEW.<c> NOTNULL AND NOT ST_IsEmpty(NEW.<c>))
BEGIN
    DELETE FROM rtree_<t>_<c> WHERE id = OLD.rowid;
    INSERT OR REPLACE INTO rtree_<t>_<c> VALUES (
      NEW.rowid,
      ST_MinX(NEW.<c>), ST_MaxX(NEW.<c>),
      ST_MinY(NEW.<c>), ST_MaxY(NEW.<c>)
    );
END;

/* Conditions: Update of any column
               Row ID change
               Empty geometry
   Actions   : Remove record from rtree for old and new rowid */
CREATE TRIGGER rtree_<t>_<c>_update4 AFTER UPDATE ON <t>
  WHEN OLD.rowid != NEW.rowid AND
```

Field Cod

```
      (NEW.<c> ISNULL OR ST_IsEmpty(NEW.<c>))
BEGIN
    DELETE FROM rtree_<t>_<c> WHERE id IN (OLD.rowid, NEW.rowid);
END;

/* Conditions: Row deleted
   Actions   : Remove record from rtree for old rowid */
CREATE TRIGGER rtree_<t>_<c>_delete AFTER DELETE ON <t>
WHEN old.<c> NOT NULL
BEGIN
   DELETE FROM rtree_<t>_<c> WHERE id = OLD.rowid;
END;
```

where <t> and <c> are replaced with the names of the feature table and geometry column being indexed.

Field Cod

Annex F Trigger Definition SQL (Informative)

F.1 geometry_columns

Table 40 gpkg_geometry_columns Trigger Definitions

```
CREATE TRIGGER 'gpkg_geometry_columns_geometry_type_name_insert'
BEFORE INSERT ON 'gpkg_geometry_columns'
FOR EACH ROW BEGIN
SELECT RAISE(ROLLBACK,'geometry_type name must be one of GEOMETRY,
POINT, LINESTRING, POLYGON, MULTIPOINT, MULTILINESTRING, MULTIPOLYGON,
GEOMCOLLECTION, CIRCULARSTRING, COMPOUNDCURVE, CURVEPOLYGON, MULTICURVE,
MULTISURFACE, CURVE, SURFACE')
WHERE NOT(NEW.geometry_type_name IN('GEOMETRY', 'POINT', 'LINESTRING',
'POLYGON', 'MULTIPOINT', 'MULTILINESTRING', 'MULTIPOLYGON',
'GEOMCOLLECTION', 'CIRCULARSTRING', 'COMPOUNDCURVE', 'CURVEPOLYGON',
'MULTICURVE', 'MULTISURFACE', 'CURVE', 'SURFACE'));
END

CREATE TRIGGER 'gpkg_geometry_columns_geometry_type_name_update'
BEFORE UPDATE OF 'geometry_type_name' ON 'gpkg_geometry_columns'
FOR EACH ROW BEGIN
SELECT RAISE(ROLLBACK,'geometry_type name must be one of GEOMETRY,
POINT, LINESTRING, POLYGON, MULTIPOINT, MULTILINESTRING, MULTIPOLYGON,
GEOMCOLLECTION, CIRCULARSTRING, COMPOUNDCURVE, CURVEPOLYGON, MULTICURVE,
MULTISURFACE, CURVE, SURFACE ')
WHERE NOT(NEW.geometry_type_name IN ('GEOMETRY', 'POINT', 'LINESTRING',
'POLYGON', 'MULTIPOINT', 'MULTILINESTRING', 'MULTIPOLYGON',
'GEOMCOLLECTION', 'CIRCULARSTRING', 'COMPOUNDCURVE', 'CURVEPOLYGON',
'MULTICURVE', 'MULTISURFACE', 'CURVE', 'SURFACE'));
END
```

F.2 gpkg_tile_matrix_metadata

Table 41 gpkg_tile_matrix_metadata Trigger Definition SQL

```
CREATE TRIGGER 'gpkg_tile_matrix_metadata_zoom_level_insert'
BEFORE INSERT ON 'gpkg_tile_matrix_metadata'
FOR EACH ROW BEGIN
SELECT RAISE(ROLLBACK, 'insert on table
''gpkg_tile_matrix_metadata'' violates constraint: zoom_level cannot
be less than 0')
WHERE (NEW.zoom_level < 0);
END

CREATE TRIGGER 'gpkg_tile_matrix_metadata_zoom_level_update'
BEFORE UPDATE of zoom_level ON 'gpkg_tile_matrix_metadata'
FOR EACH ROW BEGIN
SELECT RAISE(ROLLBACK, 'update on table
''gpkg_tile_matrix_metadata'' violates constraint: zoom_level cannot
```

Field Cod

```
be less than 0')
WHERE (NEW.zoom_level < 0);
END

CREATE TRIGGER 'gpkg_tile_matrix_metadata_matrix_width_insert'
BEFORE INSERT ON 'gpkg_tile_matrix_metadata'
FOR EACH ROW BEGIN
SELECT RAISE(ROLLBACK, 'insert on table
''gpkg_tile_matrix_metadata'' violates constraint: matrix_width
cannot be less than 1')
WHERE (NEW.matrix_width < 1);
END

CREATE TRIGGER 'gpkg_tile_matrix_metadata_matrix_width_update'
BEFORE UPDATE OF matrix_width ON 'gpkg_tile_matrix_metadata'
FOR EACH ROW BEGIN
SELECT RAISE(ROLLBACK, 'update on table
''gpkg_tile_matrix_metadata'' violates constraint: matrix_width
cannot be less than 1')
WHERE (NEW.matrix_width < 1);
END

CREATE TRIGGER 'gpkg_tile_matrix_metadata_matrix_height_insert'
BEFORE INSERT ON 'gpkg_tile_matrix_metadata'
FOR EACH ROW BEGIN
SELECT RAISE(ROLLBACK, 'insert on table
''gpkg_tile_matrix_metadata'' violates constraint: matrix_height
cannot be less than 1')
WHERE (NEW.matrix_height < 1);
END

CREATE TRIGGER 'gpkg_tile_matrix_metadata_matrix_height_update'
BEFORE UPDATE OF matrix_height ON 'gpkg_tile_matrix_metadata'
FOR EACH ROW BEGIN
SELECT RAISE(ROLLBACK, 'update on table
''gpkg_tile_matrix_metadata'' violates constraint: matrix_height
cannot be less than 1')
WHERE (NEW.matrix_height < 1);
END

CREATE TRIGGER 'gpkg_tile_matrix_metadata_pixel_x_size_insert'
BEFORE INSERT ON 'gpkg_tile_matrix_metadata'
FOR EACH ROW BEGIN
SELECT RAISE(ROLLBACK, 'insert on table
''gpkg_tile_matrix_metadata'' violates constraint: pixel_x_size must
be greater than 0')
WHERE NOT (NEW.pixel_x_size > 0);
END

CREATE TRIGGER 'gpkg_tile_matrix_metadata_pixel_x_size_update'
BEFORE UPDATE OF pixel_x_size ON 'gpkg_tile_matrix_metadata'
```

Field Cod

```
FOR EACH ROW BEGIN
SELECT RAISE(ROLLBACK, 'update on table
''gpkg_tile_matrix_metadata'' violates constraint: pixel_x_size must
be greater than 0')
WHERE NOT (NEW.pixel_x_size > 0);
END

CREATE TRIGGER 'gpkg_tile_matrix_metadata_pixel_y_size_insert'
BEFORE INSERT ON 'gpkg_tile_matrix_metadata'
FOR EACH ROW BEGIN
SELECT RAISE(ROLLBACK, 'insert on table
''gpkg_tile_matrix_metadata'' violates constraint: pixel_y_size must
be greater than 0')
WHERE NOT (NEW.pixel_y_size > 0);
END

CREATE TRIGGER 'gpkg_tile_matrix_metadata_pixel_y_size_update'
BEFORE UPDATE OF pixel_y_size ON 'gpkg_tile_matrix_metadata'
FOR EACH ROW BEGIN
SELECT RAISE(ROLLBACK, 'update on table
''gpkg_tile_matrix_metadata'' violates constraint: pixel_y_size must
be greater than 0')
WHERE NOT (NEW.pixel_y_size > 0);
END
```

F.3 metadata

Table 42 metadata Trigger Definition SQL

```
CREATE TRIGGER 'gpkg_metadata_md_scope_insert'
BEFORE INSERT ON 'gpkg_metadata'
FOR EACH ROW BEGIN
SELECT RAISE(ROLLBACK, 'insert on table gpkg_metadata violates
constraint: md_scope must be one of undefined | fieldSession |
collectionSession | series | dataset | featureType | feature |
attributeType | attribute | tile | model | catalogue | schema |
taxonomy  software | service | collectionHardware |
nonGeographicDataset | dimensionGroup')
WHERE NOT(NEW.md_scope IN
('undefined','fieldSession','collectionSession','series','dataset',
'featureType','feature','attributeType','attribute','tile','model',
'catalogue','schema','taxonomy','software','service',
'collectionHardware','nonGeographicDataset','dimensionGroup'));
END

CREATE TRIGGER 'gpkg_metadata_md_scope_update'
BEFORE UPDATE OF 'md_scope' ON 'gpkg_metadata'
FOR EACH ROW BEGIN
SELECT RAISE(ROLLBACK, 'update on table gpkg_metadata violates
constraint: md_scope must be one of undefined | fieldSession |
collectionSession | series | dataset | featureType | feature |
attributeType | attribute | tile | model | catalogue | schema |
```

Field Cod

```
taxonomy  software | service | collectionHardware |
nonGeographicDataset | dimensionGroup')
WHERE NOT(NEW.md_scope IN
('undefined','fieldSession','collectionSession','series','dataset',
'featureType','feature','attributeType','attribute','tile','model',
'catalogue','schema','taxonomy','software','service',
'collectionHardware','nonGeographicDataset','dimensionGroup'));
END
```

F.4 metadata_reference

Table 43 gpkg_metadata_reference Trigger Definition SQL

```
CREATE TRIGGER 'gpkg_metadata_reference_reference_scope_insert'
BEFORE INSERT ON 'gpkg_metadata_reference'
FOR EACH ROW BEGIN
SELECT RAISE(ROLLBACK, 'insert on table gpkg_metadata_reference
violates constraint: reference_scope must be one of "geopackage",
table", "column", "row", "row/col"')
WHERE NOT NEW.reference_scope IN
('geopackage','table','column','row','row/col');
END

CREATE TRIGGER 'gpkg_metadata_reference_reference_scope_update'
BEFORE UPDATE OF 'reference_scope' ON 'gpkg_metadata_reference'
FOR EACH ROW BEGIN
SELECT RAISE(ROLLBACK, 'update on table gpkg_metadata_reference
violates constraint: referrence_scope must be one of "geopackage",
"table", "column", "row", "row/col"')
WHERE NOT NEW.reference_scope IN
('geopackage','table','column','row','row/col');
END

CREATE TRIGGER 'gpkg_metadata_reference_column_name_insert'
BEFORE INSERT ON 'gpkg_metadata_reference'
FOR EACH ROW BEGIN
SELECT RAISE(ROLLBACK, 'insert on table gpkg_metadata_reference
violates constraint: column name must be NULL  when reference_scope
is "geopackage", "table" or "row"')
WHERE (NEW.reference_scope IN ('geopackage','table','row')
AND NEW.column_name IS NOT NULL);
SELECT RAISE(ROLLBACK, 'insert on table gpkg_metadata_reference
violates constraint: column name must be defined for the specified
table when reference_scope is "column" or "row/col"')
WHERE (NEW.reference_scope IN ('column','row/col')
AND NOT NEW.table_name IN (
SELECT name FROM SQLITE_MASTER WHERE type = 'table'
AND name = NEW.table_name
AND sql LIKE ('%' || NEW.column_name || '%')));
END

CREATE TRIGGER 'gpkg_metadata_reference_column_name_update'
```

Field Cod

```
BEFORE UPDATE OF column_name ON 'gpkg_metadata_reference'
FOR EACH ROW BEGIN
SELECT RAISE(ROLLBACK, 'update on table gpkg_metadata_reference
violates constraint: column name must be NULL when reference_scope
is "geopackage", "table" or "row"')
WHERE (NEW.reference_scope IN ('geopackage','table','row')
AND NEW.column_nameIS NOT NULL);
SELECT RAISE(ROLLBACK, 'update on table gpkg_metadata_reference
violates constraint: column name must be defined for the specified
table when reference_scope is "column" or "row/col"')
WHERE (NEW.reference_scope IN ('column','row/col')
AND NOT NEW.table_name IN (
SELECT name FROM SQLITE_MASTER WHERE type = 'table'
AND name = NEW.table_name
AND sql LIKE ('%' || NEW.column_name || '%')));
END

CREATE TRIGGER 'gpkg_metadata_reference_row_id_value_insert'
BEFORE INSERT ON 'gpkg_metadata_reference'
FOR EACH ROW BEGIN
SELECT RAISE(ROLLBACK, 'insert on table gpkg_metadata_reference
violates constraint: row_id_value must be NULL when reference_scope
is "geopackage", "table" or "column"')
WHERE NEW.reference_scope IN ('geopackage','table','column')
AND NEW.row_id_value IS NOT NULL;
SELECT RAISE(ROLLBACK, 'insert on table gpkg_metadata_reference
violates constraint: row_id_value must exist in specified table when
reference_scope is "row" or "row/col"')
WHERE NEW.reference_scope IN ('row','row/col')
  AND NOT EXISTS (SELECT rowid
   FROM (SELECT NEW.table_name AS table_name) WHERE rowid =
NEW.row_id_value);
END

CREATE TRIGGER 'gpkg_metadata_reference_row_id_value_update'
BEFORE UPDATE OF 'row_id_value' ON 'gpkg_metadata_reference'
FOR EACH ROW BEGIN
SELECT RAISE(ROLLBACK, 'update on table gpkg_metadata_reference
violates constraint: row_id_value must be NULL when reference_scope
is "geopackage", "table" or "column"')
WHERE NEW.reference_scope IN ('geopackage','table','column')
AND NEW.row_id_value  IS NOT NULL;
SELECT RAISE(ROLLBACK, 'update on table gpkg_metadata_reference
violates constraint: row_id_value must exist in specified table when
reference_scope is "row" or "row/col"')
WHERE NEW.reference_scope IN ('row','row/col')
  AND NOT EXISTS (SELECT rowid
   FROM (SELECT NEW.table_name AS table_name) WHERE rowid =
NEW.row_id_value);
END
```

Field Cod

```
CREATE TRIGGER 'gpkg_metadata_reference_timestamp_insert'
BEFORE INSERT ON 'gpkg_metadata_reference'
FOR EACH ROW BEGIN
SELECT RAISE(ROLLBACK, 'insert on table gpkg_metadata_reference
violates constraint: timestamp must be a valid time in ISO 8601
"yyyy-mm-ddThh-mm-ss.cccZ" form')
WHERE NOT (NEW.timestamp GLOB
'[1-2][0-9][0-9][0-9]-[0-1][0-9]-[1-3][0-9]T[0-2][0-9]:[0-5][0-
9]:[0-5][0-9].[0-9][0-9][0-9]Z'
AND strftime('%s',NEW.timestamp) NOT NULL);
END

CREATE TRIGGER 'gpkg_metadata_reference_timestamp_update'
BEFORE UPDATE OF 'timestamp' ON 'gpkg_metadata_reference'
FOR EACH ROW BEGIN
SELECT RAISE(ROLLBACK, 'update on table gpkg_metadata_reference
violates constraint: timestamp must be a valid time in ISO 8601
"yyyy-mm-ddThh-mm-ss.cccZ" form')
WHERE NOT (NEW.timestamp GLOB
'[1-2][0-9][0-9][0-9]-[0-1][0-9]-[1-3][0-9]T[0-2][0-9]:[0-5][0-
9]:[0-5][0-9].[0-9][0-9][0-9]Z'
AND strftime('%s',NEW.timestamp) NOT NULL);
END
```

F.5 sample_feature_table

Table 44 EXAMPLE: features table Trigger Definition SQL

```
CREATE TRIGGER "sample_feature_table_real_insert"
BEFORE INSERT ON "sample_feature_table"
FOR EACH ROW BEGIN
SELECT RAISE(ROLLBACK, 'insert on table ''sample_feature_table''
violates constraint:  real_attribute must be greater than 0')
WHERE NOT (NEW.real_attribute > 0);
END

CREATE TRIGGER "sample_feature_table_real_update"
BEFORE UPDATE OF "real_attribute" ON "sample_feature_table”
FOR EACH ROW BEGIN
SELECT RAISE (ROLLBACK, 'update of ''real_attribute'' on table
''sample_feature_table'' violates constraint: real_attribute value
must be > 0')
WHERE NOT (NEW.real_attribute > 0);
END
```

where <t> and <c> are replaced with the names of the feature table and geometry column being inserted or updated.

F.6 sample_matrix_tiles

Field Cod

Table 45 EXAMPLE: tiles table Trigger Definition SQL

```
CREATE TRIGGER "sample_matrix_tiles_zoom_insert"
BEFORE INSERT ON "sample_matrix_tiles"
FOR EACH ROW BEGIN
SELECT RAISE(ROLLBACK, 'insert on table ''sample_matrix_tiles''
violates constraint: zoom_level not specified for table in
gpkg_tile_matrix_metadata')
WHERE NOT (NEW.zoom_level IN (SELECT zoom_level FROM
gpkg_tile_matrix_metadata WHERE t_table_name =
'sample_matrix_tiles')) ;
END

CREATE TRIGGER "sample_matrix_tiles_zoom_update"
BEFORE UPDATE OF zoom_level ON "sample_matrix_tiles"
FOR EACH ROW BEGIN
SELECT RAISE(ROLLBACK, 'update on table ''sample_matrix_tiles''
violates constraint: zoom_level not specified for table in
gpkg_tile_matrix_metadata')
WHERE NOT (NEW.zoom_level IN (SELECT zoom_level FROM
gpkg_tile_matrix_metadata WHERE t_table_name =
'sample_matrix_tiles')) ;
END

CREATE TRIGGER "sample_matrix_tiles_tile_column_insert"
BEFORE INSERT ON "sample_matrix_tiles"
FOR EACH ROW BEGIN
SELECT RAISE(ROLLBACK, 'insert on table ''sample_matrix_tiles''
violates constraint: tile_column cannot be < 0')
WHERE (NEW.tile_column < 0) ;
SELECT RAISE(ROLLBACK, 'insert on table ''sample_matrix_tiles''
violates constraint: tile_column must by < matrix_width specified
for table and zoom level in gpkg_tile_matrix_metadata')
WHERE NOT (NEW.tile_column < (SELECT matrix_width FROM
gpkg_tile_matrix_metadata WHERE t_table_name = 'sample_matrix_tiles'
AND zoom_level = NEW.zoom_level));
END

CREATE TRIGGER "sample_matrix_tiles_tile_column_update"
BEFORE UPDATE OF tile_column ON "sample_matrix_tiles"
FOR EACH ROW BEGIN
SELECT RAISE(ROLLBACK, 'update on table ''sample_matrix_tiles''
violates constraint: tile_column cannot be < 0')
WHERE (NEW.tile_column < 0) ;
SELECT RAISE(ROLLBACK, 'update on table ''sample_matrix_tiles''
violates constraint: tile_column must by < matrix_width specified
for table and zoom level in gpkg_tile_matrix_metadata')
WHERE NOT (NEW.tile_column < (SELECT matrix_width FROM
gpkg_tile_matrix_metadata WHERE t_table_name = 'sample_matrix_tiles'
AND zoom_level = NEW.zoom_level));
END
```

Field Cod

```
CREATE TRIGGER "sample_matrix_tiles_tile_row_insert"
BEFORE INSERT ON "sample_matrix_tiles"
FOR EACH ROW BEGIN
SELECT RAISE(ROLLBACK, 'insert on table ''sample_matrix_tiles''
violates constraint: tile_row cannot be < 0')
WHERE (NEW.tile_row < 0) ;
SELECT RAISE(ROLLBACK, 'insert on table ''sample_matrix_tiles''
violates constraint: tile_row must by < matrix_height specified for
table and zoom level in gpkg_tile_matrix_metadata')
WHERE NOT (NEW.tile_row < (SELECT matrix_height FROM
gpkg_tile_matrix_metadata WHERE t_table_name = 'sample_matrix_tiles'
AND zoom_level = NEW.zoom_level));
END

CREATE TRIGGER "sample_matrix_tiles_tile_row_update"
BEFORE UPDATE OF tile_row ON "sample_matrix_tiles"
FOR EACH ROW BEGIN
SELECT RAISE(ROLLBACK, 'update on table ''sample_matrix_tiles''
violates constraint: tile_row cannot be < 0')
WHERE (NEW.tile_row < 0) ;
SELECT RAISE(ROLLBACK, 'update on table ''sample_matrix_tiles''
violates constraint: tile_row must by < matrix_height specified for
table and zoom level in gpkg_tile_matrix_metadata')
WHERE NOT (NEW.tile_row < (SELECT matrix_height FROM
gpkg_tile_matrix_metadata WHERE t_table_name = 'sample_matrix_tiles'
AND zoom_level = NEW.zoom_level));
END
```

Field Cod

Annex G Geometry Types (Normative)

Table 46 Geometry Type Codes (Core)

CODE	NAME
0	GEOMETRY
1	POINT
2	LINESTRING
3	POLYGON
4	MULTIPOINT
5	MULTILINESTRING
6	MULTIPOLYGON
7	GEOMCOLLECTION

Table 47 Geometry Type Codes (Extension)

CODE	NAME
8	CIRCULARSTRING
9	COMPOUNDCURVE
10	CURVEPOLYGON
11	MULTICURVE
12	MULTISURFACE
13	CURVE
14	SURFACE

GEOMETRY subtypes are POINT, CURVE, SURFACE and GEOMCOLLECTION.

CURVE subtypes are LINESTRING, CIRCULARSTRING and COMPOUNDCURVE.

Field Cod

SURFACE subtype is CURVEPOLYGON.

CURVEPOLYGON subtype is POLYGON.

GEOMCOLLECTION subtypes are MULTIPOINT, MULTICURVE and MULTISURFACE.

MULTICURVE subtype is MULTILINESTRING.

MULTISURFACE subtype is MULTIPOLYGON

Field Cod

Annex H Other SQL/MM Functions (Informative)

H.1 Geometry Routines (Informative)

Table 48 Geometry Routines

Routine Name	**Description**	**13249-3 Clause**	**99-049 Clause**	**06-103r4 Clause**	**06-104r4 Clause**
AddGeometryColumn	adds a geometry column to a feature table		2.3.8		
DropGeometryColumn	drops a geometry column from a feature table		2.3.8		
ST_Dimension	returns the dimension of a Geometry value	4.1.1.1	2.1.1.1	6.1.2.2	7.2.8.1
ST_GeometryType	returns the type of the Geometry value as a String	4.1.1.1	2.1.1.1	6.1.2.2	7.2.8.1
ST_SRID	returns the spatial reference system identifier of an Geometry value	4.1.1.1	2.1.1.1	6.1.2.2	7.2.8.1
ST_Transform	returns the Geometry value in the specified spatial reference system	4.1.1.1			
ST_IsEmpty	tests if a Geometry value corresponds to the empty set	4.1.1.1	2.1.1.1	6.1.2.2	7.2.8.1
ST_IsSimple	tests if a Geometry value has no anomalous geometric point, such as self intersection or self tangency	4.1.1.1	2.1.1.1	6.1.2.2	7.2.8.1
ST_IsValid	tests if a Geometry value is well formed	4.1.1.1			
ST_Is3D	tests whether a Geometry value has z coordinates	4.1.1.1		6.1.2.2	
ST_IsMeasured	tests whether a Geometry value has m coordinate values	4.1.1.1		6.1.2.2	
ST_LocateAlong	returns a derived geometry collection value that matches the specified m coordinate value	4.1.1.1		6.1.2.6	
ST_LocateBetween	returns a derived geometry collection value that matches the specified range of m coordinate values inclusively	4.1.1.1		6.1.2.6	

Field Cod

ST_Boundary	returns the boundary of a Geometry value	4.1.1.1	2.1.1.1	6.1.2.2	7.2.8.1
ST_Envelope	returns the bounding rectangle of a Geometry value	4.1.1.1	2.1.1.1	6.1.2.2	7.2.8.1
ST_ConvexHull	returns the convex hull of a Geometry value	4.1.1.1	2.1.1.3	6.1.2.4	7.2.8.1
ST_Buffer	returns the Geometry value that represents all points whose distance from any point of a Geometry value is less than or equal to a specified distance	4.1.1.1	2.1.1.3	6.1.2.4	7.2.8.1
ST_Intersection	returns the Geometry value that represents the point set intersection of two Geometry values	4.1.1.1	2.1.1.3	6.1.2.4	7.2.8.1
ST_Union	returns the Geometry value that represents the point set union of two ST_Geometry values	4.1.1.1	2.1.1.3	6.1.2.4	7.2.8.1
ST_Difference	returns the Geometry value that represents the point set difference of two Geometry values	4.1.1.1	2.1.1.3	6.1.2.4	7.2.8.1
ST_SymDifference	returns the Geometry value that represents the point set symmetric difference of two Geometry values	4.1.1.1	2.1.1.3	6.1.2.4	7.2.8.1
ST_Distance	returns the distance between two geometries	4.1.1.1	2.1.1.3	6.1.2.4	7.2.8.1
ST_AsText	returns the well-known text representation for the specified Geometry value	4.1.1.1	2.1.1.1	6.1.2.2	7.2.8.1
ST_AsBinary	returns the well-known binary representation for the specified Geometry value	4.1.1.1	2.1.1.1	6.1.2.2	
ST_AsGML	returns the GML representation for the specified Geometry value	4.1.1.1			
ST_GeomFromText	returns a Geometry value from its well-known text representation	4.1.1.2	3.2.6.2		
ST_GeomFromWKB	returns a Geometry value from its well-known binary representation	4.1.1.2	3.2.7.2		

Field Cod

ST_GeomFromGML	returns a Geometry value from its GML representation	4.1.1.2			

Table 49 Point Routines

Routine Name	Description	13249-3 Clause	99-049 Clause	06-103r4 Clause	06-104r4 Clause
ST_X	returns the x coordinate value of a Point value	4.1.3.1	3.2.11.2	6.1.4.2	7.2.9.1
ST_Y	returns the y coordinate value of a Point value	4.1.3.1	3.2.11.2	6.1.4.2	7.2.9.1
ST_Z	returns the z coordinate value of a Point value	4.1.3.1		6.1.4.2	7.2.9.1
ST_M	returns the m coordinate value of a Point value	4.1.3.1		6.1.4.2	7.2.9.1
ST_PointFromText	returns a Point value from the well-known text representation of a Point	4.1.3.2	3.2.6.2		
ST_PointFromWKB	returns a Point value from the well-known binary representation of a Point.	4.1.3.2	3.2.7.2		

Table 50 Curve Routines

Routine Name	Description	13249-3 Clause	99-049 Clause	06-103r4 Clause	06-104r4 Clause
ST_Length	returns the length of a Curve value	4.1.4.1	2.1.5.1	6.1.6.2	7.2.10.1
ST_StartPoint	returns the Point value that is the start point of a Curve value	4.1.4.1	2.1.5.1	6.1.6.2	7.2.10.1
ST_EndPoint	returns the Point value that is the end point of a Curve value	4.1.4.1	2.1.5.1	6.1.6.2	7.2.10.1
ST_IsClosed	tests if a Curve value is closed	4.1.4.1	2.1.5.1	6.1.6.2	7.2.10.1
ST_IsRing	tests if an Curve value is a ring	4.1.4.1	2.1.5.1	6.1.6.2	7.2.10.1

Table 51 LineString Routines

Routine Name	Description	13249-3 Clause	99-049 Clause	06-103r4 Clause	06-104r4 Clause
ST_NumPoints	returns the cardinality of the Point collection in the LineString value	4.1.5.1	2.1.6.1	6.1.7.2	7.2.11.1

ST_PointN	returns the specified element in the Point collection in the LineString value	4.1.5.1	2.1.6.1	6.1.7.2	7.2.11.1
ST_LineFromText	returns a LineString value from the well-known text representation of a LineString	4.1.5.2	3.2.6.2		
ST_LineFromWKB	returns a LineString value from the well-known binary representation of a LineString	4.1.5.2	3.2.7.2		

Table 52 Surface Routines

Routine Name	**Description**	**13249-3 Clause**	**99-049 Clause**	**06-103r4 Clause**	**06-104r4 Clause**
ST_Area	returns the area of a Surface value	4.1.8.1	2.1.9.1	6.1.10.2	7.2.12.1
ST_Centroid	returns the Point value that is the mathematical centroid of the Surface value	4.1.8.1	2.1.9.1	6.1.10.2	7.2.12.1
ST_PointOnSurface	returns a Point value that is guaranteed to be on the Surface value	4.1.8.1	2.1.9.1	6.1.10.2	7.2.12.1

Table 53 Polygon Routines

Routine Name	**Description**	**13249-3 Clause**	**99-049 Clause**	**06-103r4 Clause**	**06-104r4 Clause**
ST_ExteriorRing	returns the exterior ring of a Polygon value	4.1.9.1	2.1.10.1	6.1.11.2	7.2.13.1
ST_NumInteriorRing	returns the cardinality of the collection of interior rings of a Polygon value	4.1.9.1	2.1.10.1	6.1.11.2	7.2.13.1
ST_InteriorRingN	returns the specified element in the collection of interior rings of a Polygon value	4.1.9.1	2.1.10.1	6.1.11.2	7.2.13.1
ST_PolyFromText	returns a Polygon value from the well-known text representation of a Polygon	4.1.10.2	3.2.6.2		
ST_PolyFromWKB	returns a Polygon value from the well-known binary representation of a Polygon	4.1.10.2	3.2.7.2		

Field Cod

ST_BdPolyFromText	returns a Polygon value from a well-known text representation of a MultiLineString	4.1.10.2	3.2.6.2		
ST_BdPolyFromWKB	returns a Polygon value from a well-known binary representation of a MultiLineString	4.1.10.2	3.2.7.2		

Table 54 MultiGeometry Routines

Routine Name	**Description**	**13249-3 Clause**	**99-049 Clause**	**06-103r4 Clause**	**06-104r4 Clause**
ST_NumGeometries	returns the cardinality of the Geometry collection	4.1.11.1	3.2.16.2	6.1.3.2	7.2.15
ST_GeometryN	returns the specified element in the Geometry collection	4.1.11.1	3.2.16.2	6.1.3.2	7.2.15
ST_GeomCollFromTxt	returns a GeomCollection value from the well-known text representation of a GeomCollection	4.1.11.2	3.2.6.2		
ST_GeomCollFromWKB	returns a GeomCollection value from the well-known binary representation of GeomCollection	4.1.11.2	3.2.7.2		

Table 55 MultiPoint Routines

Routine Name	**Description**	**13249-3 Clause**	**99-049 Clause**	**06-103r4 Clause**	**06-104r4 Clause**
ST_MPointFromText	returns a MultiPoint value from the well-known text representation of a MultiPoint	4.1.12.2	3.2.6.2		
ST_MPointFromWKB	returns a MultiPoint value from the well-known binary representation of a MultiPoint	4.1.12.2	3.2.7.2		

Table 56 MultiCurve Routines

Routine Name	**Description**	**13249-3 Clause**	**99-049 Clause**	**06-103r4 Clause**	**06-104r4 Clause**

Field Cod

ST_IsClosed	tests if a MultiCurve value is closed	4.1.13.1	3.2.17.2	6.1.8.2	7.2.17.1
ST_Length	returns the length of a MultiCurve value	4.1.13.1	3.2.17.2	6.1.8.2	7.2.17.1

Table 57 MultiLineString Routines

Routine Name	Description	13249-3 Clause	99-049 Clause	06-103r4 Clause	06-104r4 Clause
ST_MLineFromText	returns a MultiLineString value from the well-known text representation of a MultiLineString	4.1.14.2	3.2.6.2		
ST_MLineFromWKB	returns a MultiLineString value from the well-known binary representation of a MultiLineString	4.1.14.2	3.2.7.2		

Table 58 MultiSurface Routines

Routine Name	Description	13249-3 Clause	99-049 Clause	06-103r4 Clause	06-104r4 Clause
ST_Area	returns the area of a MultiSurface value	4.1.15.1	3.2.18.2	6.1.13.2	7.2.19.1
ST_Centroid	returns the Point value that is the mathematical centroid of the MultiSurface value	4.1.15.1	3.2.18.2	6.1.13.2	7.2.19.1
ST_PointOnSurface	returns a Point value that is guaranteed to be on the MultiSurface value	4.1.15.1	3.2.18.2	6.1.13.2	7.2.19.1

Table 59 MultiPolygon Routines

Routine Name	Description	13249-3 Clause	99-049 Clause	06-103r4 Clause	06-104r4 Clause
ST_MPolyFromText	returns a MultiPolygon value from the well-known text representation of a MultiPolygon	4.1.16.2	3.2.6.2		
ST_MPolyFromWKB	returns a MultiPolygon value from the well-known binary representation of a MultiPolygon	4.1.16.2	3.2.7.2		
ST_BdMPolyFromText	returns a MultiPolygon value from a well-known text representation of a MultiLineString	4.1.16.2	3.2.6.2		

Field Cod

ST_BdMPolyFromWKB	returns a MultiPolygon value from a well-known binary representation of a MultiLineString	4.1.16.2	3.2.7.2		

Table 60 Spatial Predicates

Routine Name	**Description**	**13249-3 Clause**	**99-049 Clause**	**06-103r4 Clause**	**06-104r4 Clause**
ST_Equals	tests if a Geometry value is spatially equal to another Geometry value.	4.1.2.3	3.2.19.2	6.1.15.3	7.2.8.1
ST_Relate	tests if a Geometry value is spatially related to another Geometry value by testing for intersections between the interior, boundary and exterior of the two Geometry values as specified by the intersection matrix.	4.1.2.3	3.2.19.2	6.1.15.3	7.2.8.1
ST_Disjoint	tests if a Geometry value is spatially disjoint from another Geometry value	4.1.2.3	3.2.19.2	6.1.15.3	7.2.8.1
ST_Intersects	tests if a Geometry value spatially intersects another Geometry value	4.1.2.3	3.2.19.2	6.1.15.3	7.2.8.1
ST_Touches	tests if a Geometry value spatially touches another Geometry value	4.1.2.3	3.2.19.2	6.1.15.3	7.2.8.1
ST_Crosses	tests if a Geometry value spatially crosses another Geometry value	4.1.2.3	3.2.19.2	6.1.15.3	7.2.8.1
ST_Within	tests if a Geometry value is spatially within another Geometry value	4.1.2.3	3.2.19.2	6.1.15.3	7.2.8.1
ST_Contains	tests if a Geometry value spatially contains another Geometry value	4.1.2.3	3.2.19.2	6.1.15.3	7.2.8.1
ST_Overlaps	tests if a Geometry value spatially overlaps another Geometry value	4.1.2.3	3.2.19.2	6.1.15.3	7.2.8.1

Field Cod

Annex I Hierarchical Metadata Exammple (Informative)

The first example use case is from ISO19115 H.2.

Suppose we have this metadata:

```
CREATE TABLE gpkg_metadata (
  id INTEGER NOT NULL PRIMARY KEY,
  md_scope TEXT NOT NULL DEFAULT 'undefined',

  metadata_standard_URI TEXT NOT NULL DEFAULT
'http://schemas.opengis.net/iso/19139/',
  metadata BLOB NOT NULL DEFAULT (zeroblob(4))
)
```

id	md_scope	metadata_standard_uri	metadata
0	undefined	http://schemas.opengis.net/iso/19139/	BLOB
3	series	http://schemas.opengis.net/iso/19139/	BLOB
4	dataset	http://schemas.opengis.net/iso/19139/	BLOB
5	featureType	http://schemas.opengis.net/iso/19139/	BLOB
6	feature	http://schemas.opengis.net/iso/19139/	BLOB
7	attributeType	http://schemas.opengis.net/iso/19139/	BLOB
8	attribute	http://schemas.opengis.net/iso/19139/	BLOB

and this reference table definition:

```
CREATE TABLE gpkg_metadata_reference (
  reference_scope TEXT NOT NULL DEFAULT "table",
  table_name TEXT NOT NULL DEFAULT "undefined",
  column_name TEXT NOT NULL DEFAULT "undefined",
  row_id_value INTEGER NOT NULL DEFAULT 0,
  timestamp TEXT NOT NULL DEFAULT (strftime('%Y-%m-
%dT%H:%M:%fZ',CURRENT_TIMESTAMP)),
  md_file_id INTEGER NOT NULL DEFAULT 0,
  md_parent_id INTEGER NOT NULL DEFAULT 0,
  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)
)
```

H.2 1) Consider a geographic data provider generating vector mapping data for three Administrative areas(A, B and C). ... The metadata could be carried exclusively at Dataset Series level.

Then we need a record for each layer table for the three admin areas, like this:

```
INSERT INTO gpkg_metadata_reference VALUES (
'table', /* reference type */
'roads', /* table name */
'undefined', /* column_name */
-1, /* row_id_value  */
(datetime('now')),
```

Field Cod

```
3,  /* md_file_id  */
0  /* md_parent_id  */
)
```

H.2 2) After some time alternate vector mapping of Administrative area A becomes available. The metadata would then be extended for Administrative area A, to describe the new quality date values. These values would supersede those given for the Dataset series, but only for Administrative area A. The metadata for B and C would remain unchanged. This new metadata would be recorded at Dataset
level.

Then we need a record for each layer table in "A" like this:

```
INSERT INTO gpkg_metadata_reference VALUES (
'table', /* reference type */
'roads', /* table name */
'undefined', /* column_name */
-1, /* row_id_value  */
(datetime('now')),
 4,  /* md_file_id  */
3  /* md_parent_id  */
)
```

H.2 3) Eventually further data becomes available for Administrative area A, with a complete re-survey of the road network. Again this implies new metadata for the affected feature types. This metadata would be carried at Feature type level for Administrative area A. All other metadata relating to other feature types remains unaffected. Only the metadata for roads in Administrative area A is modified. This road metadata is recorded at Feature type level.

Then we need a record for each layer table for the roads network, like this:

```
INSERT INTO gpkg_metadata_reference VALUES (
'table', /* reference type */
'roads', /* table name */
'undefined', /* column_name */
-1, /* row_id_value  */
(datetime('now')),
5,  /* md_file_id  */
4  /* md_parent_id  */
)
```

H.2 4) An anomaly in the road survey is identified, in that all Overhead clearances for the Administrative area A have been surveyed to the nearest metre. These are re-surveyed to the nearest decimetre. This re-survey implies new metadata for the affected attribute type 'Overhead Clearance'. All other metadata for Administrative area A remains unaffected. This 'Overhead Clearance' metadata is recorded at Attribute Type level

Then we need a record for each layer table in the roads network with attribute type 'Overhead Clearance', like this;

```
INSERT INTO gpkg_metadata_reference VALUES (
'column', /* reference type */
'roads', /* table name */
'overhead_clearance', /* column_name */
```

Field Cod

```
-1, /* row_id_value  */
(datetime('now')),
7,  /* md_file_id  */
4  /* md_parent_id  */
)
```

H.2 5) A new bridge is constructed in Administrative area A. This new data is reflected in the geographic data for Administrative area A, and new metadata is required to record this new feature. All other metadata for Administrative area A remains unaffected. This new feature metadata is recorded at Feature instance level.

Then we need a record for the bridge layer table row for the new bridge, like this:

```
INSERT INTO gpkg_metadata_reference VALUES (
'row', /* reference type */
'bridge', /* table name */
'undefined', /* column_name */
987, /* row_id_value  */
(datetime('now')),
6,  /* md_file_id  */
4  /* md_parent_id  */
)
```

H.2 6) The overhead clearance attribute of the new bridge was wrongly recorded, and is modified. Again this new attribute requires new metadata to describe the modification. All other metadata for Administrative area A remains unaffected. This new attribute metadata is recorded at Attribute instance level.

Then we need a record for the clearance attribute value, like this:

```
INSERT INTO gpkg_metadata_reference VALUES (
'row/col', /* reference type */
'bridge', /* table name */
'overhead_clearance', /* column_name */
987, /* row_id_value  */
(datetime('now')),
8,  /* md_file_id  */
4  /* md_parent_id  */
)
```

The second example use case is for a field data collection session. This use case demonstrates a mechanism to indicate which data in a GeoPackage that was originally loaded with data from one or more services has been collected or updated since the initial load, and therefore MAY need to be uploaded to update the original services (e.g. WFS, WCS, WMTS).

Suppose a user with a mobile handheld device goes out in the field and collects observations of a new "Point of Interest" (POI) feature type, and associated metadata about the field session, the new feature type, some POI instances and some of their attributes (e.g. spatial accuracy, attribute accuracy) that results in the following additional metadata:

id	md_scope	metadata_standard_uri	metadata
0	undefined	http://schemas.opengis.net/iso/19139/	BLOB
1	fieldSession	http://schemas.opengis.net/iso/19139/	BLOB
10	featureType	http://schemas.opengis.net/iso/19139/	BLOB

Field Cod

11	feature	http://schemas.opengis.net/iso/19139/	BLOB
12	attribute	http://schemas.opengis.net/iso/19139/	BLOB
13	attribute	http://schemas.opengis.net/iso/19139/	BLOB
14	feature	http://schemas.opengis.net/iso/19139/	BLOB
15	attribute	http://schemas.opengis.net/iso/19139/	BLOB
16	attribute	http://schemas.opengis.net/iso/19139/	BLOB
17	feature	http://schemas.opengis.net/iso/19139/	BLOB
18	attribute	http://schemas.opengis.net/iso/19139/	BLOB
19	attribute	http://schemas.opengis.net/iso/19139/	BLOB

(This example assumes that the field session data is still considered "raw" and won't be considered a data set or part of a data series until it has been verified and cleaned, but if that is wrong then additional series and data set metadata could be added.)

Then we need a gpkg_metadata_reference record for the field session for the new POI table, whose md_parent_id is undefined:

```
INSERT INTO gpkg_metadata_reference VALUES (
'table', /* reference type */
'poi', /* table name */
'undefined', /* column_name */
-1, /* row_id_value  */
(strftime('%Y-%m-%dT%H:%M:%fZ','now')),
1,  /* md_file_id  */
0  /* md_parent_id  */
)
```

Then we need a gpkg_metadata_reference record for the feature type for the new POI table, whose md_parent_id is that of the field session:

```
INSERT INTO gpkg_metadata_reference VALUES (
'table', /* reference type */
'poi', /* table name */
'undefined', /* column_name */
-1, /* row_id_value  */
(strftime('%Y-%m-%dT%H:%M:%fZ','now')),
10,  /* md_file_id  */
1  /* md_parent_id  */
)
```

Then we need gpkg_metadata_reference records for the poi feature instance rows, whose md_parent_id is that of the field session:

```
INSERT INTO gpkg_metadata_reference VALUES (
'row', /* reference type */
'poi', /* table name */
'undefined', /* column_name */
1, /* row_id_value  */
(strftime('%Y-%m-%dT%H:%M:%fZ','now')),
11,  /* md_file_id  */
1  /* md_parent_id  */
)
```

Field Cod

```
INSERT INTO gpkg_metadata_reference VALUES (
'row', /* reference type */
'poi', /* table name */
'undefined', /* column_name */
2, /* row_id_value  */
14,  /* md_file_id  */
1  /* md_parent_id  */
)
INSERT INTO gpkg_metadata_reference VALUES (
'row', /* reference type */
'poi', /* table name */
'undefined', /* column_name */
3, /* row_id_value  */
(strftime('%Y-%m-%dT%H:%M:%fZ','now')),
17,  /* md_file_id  */
1  /* md_parent_id  */
)
```

And finally we need gpkg_metadata_reference records for the poi attribute instance metadata , whose md_parent_id is that of the field session:

```
INSERT INTO gpkg_metadata_reference VALUES (
'row/col', /* reference type */
'poi', /* table name */
'point', /* column_name */
1, /* row_id_value  */
(strftime('%Y-%m-%dT%H:%M:%fZ','now')),
12,  /* md_file_id  */
1  /* md_parent_id  */
)
INSERT INTO gpkg_metadata_reference VALUES (
'row/col', /* reference type */
'poi', /* table name */
'point', /* column_name */
2, /* row_id_value  */
(strftime('%Y-%m-%dT%H:%M:%fZ','now')),
15,  /* md_file_id  */
1  /* md_parent_id  */
)
INSERT INTO gpkg_metadata_reference VALUES (
'row/col', /* reference type */
'poi', /* table name */
'point', /* column_name */
3, /* row_id_value  */
(strftime('%Y-%m-%dT%H:%M:%fZ','now')),
18,  /* md_file_id  */
1  /* md_parent_id  */
)
INSERT INTO gpkg_metadata_reference VALUES (
'row/col', /* reference type */
'poi', /* table name */
'category', /* column_name */
```

Field Cod

```
1, /* row_id_value  */
(strftime('%Y-%m-%dT%H:%M:%fZ','now')),
13,  /* md_file_id  */
1  /* md_parent_id  */
)
INSERT INTO gpkg_metadata_reference VALUES (
'row/col', /* reference type */
'poi', /* table name */
'category', /* column_name */
2, /* row_id_value  */
(strftime('%Y-%m-%dT%H:%M:%fZ','now')),
16,  /* md_file_id  */
1  /* md_parent_id  */
)
INSERT INTO gpkg_metadata_reference VALUES (
'row/col', /* reference type */
'poi', /* table name */
'category', /* column_name */
3, /* row_id_value  */
(strftime('%Y-%m-%dT%H:%M:%fZ','now')),
19,  /* md_file_id  */
1  /* md_parent_id  */
)
```

As long as all metadata collected in the field session either directly (as above) or indirectly (suppose there were a data set level metadata_reference record intermediary) refers to the field session metadata via md_parent_id values, then this chain of metadata references identifies the newly collected information, as Joan requested, in addition to the metadata.

So here is the data after both examples:

xml_metadata

id	md_scope	metadata_standard_uri	metadata
0	undefined	http://schemas.opengis.net/iso/19139/	BLOB
1	fieldSession	http://schemas.opengis.net/iso/19139/	BLOB
2	collectionSession	http://schemas.opengis.net/iso/19139/	BLOB
3	series	http://schemas.opengis.net/iso/19139/	BLOB
4	dataset	http://schemas.opengis.net/iso/19139/	BLOB
5	featureType	http://schemas.opengis.net/iso/19139/	BLOB
6	feature	http://schemas.opengis.net/iso/19139/	BLOB
7	attributeType	http://schemas.opengis.net/iso/19139/	BLOB
8	attribute	http://schemas.opengis.net/iso/19139/	BLOB
10	featureType	http://schemas.opengis.net/iso/19139/	BLOB
11	feature	http://schemas.opengis.net/iso/19139/	BLOB
12	attribute	http://schemas.opengis.net/iso/19139/	BLOB
13	attribute	http://schemas.opengis.net/iso/19139/	BLOB
14	feature	http://schemas.opengis.net/iso/19139/	BLOB
15	attribute	http://schemas.opengis.net/iso/19139/	BLOB
16	attribute	http://schemas.opengis.net/iso/19139/	BLOB

Field Cod

17	feature	http://schemas.opengis.net/iso/19139/	BLOB
18	attribute	http://schemas.opengis.net/iso/19139/	BLOB
19	attribute	http://schemas.opengis.net/iso/19139/	BLOB

gpkg_metadata_reference

reference_type	**table_name**	**column_name**	**row_id_value**	**timestamp**	**md_file_id**	**md_parent_id**
table	roads	undefined	0	ts	3	0
table	roads	undefined	0	ts	4	3
table	roads	undefined	0	ts	5	4
column	roads	overhead_clearance	0	ts	7	4
row	bridge	undefined	987	ts	6	4
row/col	bridge	overhead_clearance	987	ts	8	4
table	poi	undefined	0	ts	1	0
row	poi	undefined	0	ts	10	1
row	poi	undefined	1	ts	11	1
row	poi	undefined	2	ts	14	1
row/col	poi	undefined	3	ts	17	1
row/col	poi	point	1	ts	12	1
row/col	poi	point	2	ts	15	1
row/col	poi	point	3	ts	18	1
row/col	poi	category	1	ts	13	1
row/col	poi	category	2	ts	16	1
row/col	poi	category	3	ts	19	1

Field Cod

Annex J Raster or Tile Metadata Example (Informative)

A number of raster image processing problems MAY require the support of more metadata that is contained in the image itself. Applications MAY use the gpkg_metadata and gpkg_metadata_reference tables defined in clause 1.2.5 to store raster image metadata defined according to standard authoritative or application or vendor specific metadata models. An example of the data items in such a model is shown in the following table.

- Rational Polynomial Coefficient
- Photometric Interpretation
- No Data Value
- Compression Quality Factor
- Georectification
- NIIRS
- Min X
- Min Y
- Max X
- Max Y

Field Cod

Annex K Normative References (Normative)

The following normative documents contain provisions which, through reference in this text, constitute provisions of OGC 12-128 For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of OGC 12-128 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies.

[1] ISO 19105: Geographic information — Conformance and Testing

[2] ISO/IEC 9075:1992 Information Technology - Database Language SQL SQL92)

[3] ISO/IEC 9075-1:2011 Information Technology - Database Language SQL - Part 1: Framework

[4] ISO/IEC 9075-2:2011 Information Technology - Database Language SQL - Part 2: Foundation

[5] ISO/IEC 9075-3:2008 Information Technology - Database Language SQL - Part 3: Call-Level Interface (SQL/CLI)

[6] ISO/IEC 9075-4:2011 Information Technology - Database Language SQL - Part 4: Persistent Stored Modules (SQL/PSM)

[7] ISO/IEC 9075-10:2008 Information Technology - Database Language SQL – Part 10: Object Language Bindings (SQL/OLB)

[8] JDBC™ 3.0 Specification, Final Release, John Ellis & Linda Ho with Maydene Fisher, Sun Microsystems, Inc., October, 2001.

[9] SQLite (all parts) http://www.sqlite.org/ (online) http://www.sqlite.org/sqlite-doc-3071300.zip (offline)

[10] http://sqlite.org/fileformat2.html

[11] http://www.sqlite.org/formatchng.html

[12] http://www.sqlite.org/download.html

[13] OGC 06-103r4 OpenGIS® Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture Version: 1.2.1 2011-05-28 http://portal.opengeospatial.org/files/?artifact_id=25355 (also ISO/TC211 19125 Part 1)

[14] OGC 06-104r4 OpenGIS® Implementation Standard for Geographic information - Simple feature access - Part 2: SQL option Version: 1.2.1 2010-08-04 http://portal.opengeospatial.org/files/?artifact_id=25354 (also ISO/TC211 19125 Part 2)

[15] OGC 99-049 OpenGIS® Simple Features Specification for SQL Revision 1.1 May 5, 1999, Clause 2.3.8 http://portal.opengeospatial.org/files/?artifact_id=829

[16] ISO/IEC 13249-3:2011 Information technology — SQL Multimedia and Application Packages - Part 3: Spatial (SQL/MM)

[17] http://www.epsg.org/Geodetic.html

Field Cod

[18] http://www.epsg-registry.org/

[19] MIL_STD_2401 DoD World Geodetic System 84 (WGS84), 11 January 1994

[20] OGC 07-057r7 OpenGIS® Web Map Tile Service Implementation Standard ersion 1.0.0 2010-04-06 (WMTS) http://portal.opengeospatial.org/files/?artifact_id=35326

[21] ITU-T Recommendation T.81 (09/92) with Corrigendum (JPEG)

[22] JPEG File Interchange Format Version 1.02, September 1, 1992 http://www.jpeg.org/public/jfif.pdf

[23] IETF RFC 2046 Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types http://www.ietf.org/rfc/rfc2046.txt

[24] Portable Network Graphics http://libpng.org/pub/png/

[25] MIME Media Types http://www.iana.org/assignments/media-types/index.html

[26] WebP https://developers.google.com/speed/webp/

[27] TIFF – Tagged Image File Format, Revision 6.0, Adobe Systems Inc., June 1992 http://partners.adobe.com/public/developer/en/tiff/TIFF6.pdf

[28] GeoTIFF Format Specification, Revision 1.0, 10 November 1995; version 1.8.2 http://www.remotesensing.org/geotiff/spec/geotiffhome.html

[29] NGA Standardization Document: Implementation Profile for Tagged Image File Format (TIFF) and Geographic Tagged Image File Format (GeoTIFF), Version 2.0, 2001-10-26 https://nsgreg.nga.mil/doc/view?i=2224

[30] IETF RFC 3986 Uniform Resource Identifier (URI): Generic Syntax http://www.ietf.org/rfc/rfc3986.txt

[31] OGC08-131r3 The Specification Model — A Standard for Modular specifications https://portal.opengeospatial.org/files/?artifact_id=34762

[32] OGC10-103 Name type specification - specification elements http://portal.opengeospatial.org/files/?artifact_id=39194

[33] W3C Recommendation 26 November 2008 Extensible Markup Language (XML) 1.0 (Fifth Edition) http://www.w3.org/TR/xml/

[34] W3C Recommendation 8 December 2009 Namespaces in XML 1.0 (Third Edition) http://www.w3.org/TR/REC-xml-names/

[35] W3C Recommendation 28 January 2009 XML Base (Second Edition) http://www.w3.org/TR/xmlbase/

[36] W3C Recommendation 06 May 2010 XML Linking Language (XLink) Version 1.1 http://www.w3.org/TR/xlink11/

[37] W3C Recommendation 28 October 2004 XML Schema Part 0: Primer Second Edition http://www.w3.org/TR/xmlschema-0/

[38] W3C Recommendation 28 October 2004 XML Schema Part 1: Structures Second Edition http://www.w3.org/TR/xmlschema-1/

Field Cod

[39] W3C Recommendation 28 October 2004 XML Schema Part 2: Datatypes Second Edition http://www.w3.org/TR/xmlschema-2/

[40] ISO 19115 Geographic information -- Metadata, 8 May 2003, with Technical Corrigendum 1, 5 July 2006

[41] ISO 8601 Representation of dates and times http://www.iso.org/iso/catalogue_detail?csnumber=40874

[42] OGC® 10-100r3 Geography Markup Language (GML) simple features profile (with technical note) http://portal.opengeospatial.org/files/?artifact_id=42729

[43] Atom Syndication Format - IETF RFC 4287 http://tools.ietf.org/html/rfc4287

[44] OGC® OWS Context Documen Conceptual Model 12-080 OWS Context Conceptual Model https://portal.opengeospatial.org/wiki/OWSContextswg/ConceptualModelHome

[45] OGC® OWS Context Atom Encoding 2-084 OWS Context Atom Encoding https://portal.opengeospatial.org/wiki/OWS9/GeoPackageOWSContext

[46] MIL-STD-2500C DoD Interface Standard: National Imagery Transmission Format (NITF) https://nsgreg.nga.mil/NSGDOC/files/doc/Document/MIL-STD-2500C.pdf

Field Cod

Annex L Bibliography (Informative)

[B1] http://developer.android.com/guide/topics/data/data-storage.html#db

[B2] https://developer.apple.com/technologies/ios/data-management.html

[B3] http://www.epsg.org/guides/docs/G7-1.pdf

[B4] http://www.gdal.org/ogr/

[B5] http://www.gdal.org/ogr/drv_sqlite.html

[B6] http://trac.osgeo.org/geos/wiki/Applications

[B7] http://www.qgis.org/

[B8] http://hub.qgis.org/projects/android-qgis

[B9] http://www.luciad.com/products/LuciadMobile

[B10] http://www.falconview.org/trac/FalconView/downloads/26

[B11] http://wiki.openstreetmap.org/wiki/TMS

[B12] https://github.com/mapbox/mbtiles-spec

[B13] https://www.gaia-gis.it/fossil/librasterlite/index

[B14] http://wiki.openstreetmap.org/wiki/Main_Page

[B15] http://code.google.com/p/osmdroid/

[B16] http://www.falconview.org/trac/FalconView

[B17] http://code.google.com/p/big-planet-tracks/

[B18] http://en.wikipedia.org/wiki/ASCII

[B19] "SQL for Smarties: Advanced SQL Programming" Joe Selko, Morgan Kaufmann, 1995, ISBN 1-55860-323-9

[B20] NATO IIRS STANAG, NATO Imagery Interpretability Rating Scale (NIIRS) STANAG 7194 Edition 1 2009

[B21] https://nsgreg.nga.mil/doc/view?i=2129

[B22] U.S. National Image Interpretability Rating Scales http://www.fas.org/irp/imint/niirs.htm

[B23] Civil NIIRS http://www.fas.org/irp/imint/niirs_c/index.html

[B24] Civil NIIRS Reference Guide http://www.fas.org/irp/imint/niirs_c/guide.htm

[B25] Additional Civil NIIRS Criteria http://www.fas.org/irp/imint/niirs_c/app2.htm

[B26] Sample Civil NIIRS Images http://www.fas.org/irp/imint/niirs_c/append.htm

[B27] History of NIIRS http://www.fas.org/irp/imint/niirs_c/app3.htm

[B28] http://www.ucgis.org/priorities/research/research_white/1998%20Papers/data.html

[B29] http://www.gdal.org/frmt_rasterlite.html

Field Cod

[B30] https://www.gaia-gis.it/fossil/libspatialite/wiki?name=switching-to-4.0

[B31] http://www.sqlite.org/lang_createtable.html#rowid

[B32] ISO 19115-2 Geographic information - - Metadata - Part 2: Metadata for imagery and gridded data

[B33] ISO 19139: Geographic information -- Metadata -- XML schema implementation

[B34] Dublin Core Metadata Initiative http://dublincore.org/ IETF RFC 5013

[B35] ISO 15836:2009 http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=52142

[B36] Content Standard for Digital Geospatial Metadata (CSDGM)

[B37] http://www.fgdc.gov/standards/projects/FGDC-standards-projects/metadata/base-metadata/index_html

[B38] Department of Defense Discovery Metadata Specification (DDMS) http://metadata.ces.mil/mdr/irs/DDMS/

[B39] NMF NGA.STND.0012_2.0 / NMIS NGA.STND.0018_1.0

[B40] Unified Modeling Language (UML) http://www.uml.org/

[B41] XML for Metadata Interchange (XMI) http://www.omg.org/spec/XMI/

[B42] Geography Markup Language (GML) ISO 19136:2007

[B43] ISO 19110 Geographic information – Methodology for feature cataloguing

[B44] Web Ontology Language (OWL) http://www.w3.org/TR/2009/REC-owl2-xml-serialization-20091027/

[B45] Simple Knowledge Organization System (SKOS) http://www.w3.org/TR/skos-reference/

[B46] STANAG 7074 Digital Geographic Information Exchange Standard (DIGEST) - AGeoP-3A, edition 1, 19 October 1994
http://www.dgiwg.org/dgiwg/htm/documents/historical_documents.htm

[B47] ISO 19109 Geographic information - Rules for application schema

[B48] http://www.sqlite.org/changes.html

[B49] http://sqlite.org/src4/doc/trunk/www/design.wiki

[B50] http://trac.osgeo.org/geos/

[B51] http://trac.osgeo.org/proj

[B52] https://www.gaia-gis.it/fossil/libspatialite/index

[B53] http://www.gaia-gis.it/gaia-sins/BLOB-Geometry.html

Field Cod