

Open Geospatial Consortium Inc.

Date: 2013-11-19

Reference number of this OGC® project document: OGC 12-128r9

Version: 0.9.12

Category: OGC® Encoding Standard

Editor: Paul Daisey

OGC® GeoPackage Encoding Standard

Copyright notice

Copyright © 2012.Envitia
Copyright © 2012.Luciad
Copyright © 2012 Sigma Bravo.
Copyright © 2012 The Carbon Project.
Copyright © 2012 U.S. Army Geospatial Center.
Copyright © 2012 U.S. National Geospatial Intelligence Agency.

The companies listed above have granted the Open Geospatial Consortium, Inc. (OGC) a nonexclusive, royalty-free, paid up, worldwide license to copy and distribute this document and to modify this document and distribute copies of the modified version. To obtain additional rights of use, visit <http://www.opengeospatial.org/legal/>.

Warning

This document is not an OGC Standard. It is distributed for review and comment. It is subject to change without notice and SHALL NOT be referred to as an OGC Standard. Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

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

INTRODUCTION.....	V
1. BASE	1
1.1. CORE	1
1.1.1. <i>SQLite Container</i>	1
1.1.2. <i>Spatial Reference Systems</i>	4
1.1.3. <i>Contents</i>	5
2. OPTIONS	7
2.1. FEATURES	7
2.1.1. <i>Simple Features SQL Introduction</i>	7
2.1.2. <i>Contents</i>	9
2.1.3. <i>Geometry Encoding</i>	9
2.1.4. <i>SQL Geometry Types</i>	11
2.1.5. <i>Geometry Columns</i>	11
2.1.6. <i>Vector Feature User Data Tables</i>	13
2.2. TILES	14
2.2.1. <i>Tile Matrix Introduction</i>	14
2.2.2. <i>Contents</i>	15
2.2.3. <i>Zoom Levels</i>	15
2.2.4. <i>Tile Encoding PNG</i>	16
2.2.5. <i>Tile Encoding JPEG</i>	16
2.2.6. <i>Tile Matrix Set</i>	16
2.2.7. <i>Tile Matrix</i>	18
2.2.8. <i>Tile Pyramid User Data Tables</i>	20
2.3. SCHEMA	21
2.3.1. <i>Schema Introduction</i>	21
2.3.2. <i>Data Columns</i>	21
2.3.3. <i>Data Column Constraints</i>	22
2.4. METADATA	25
2.4.1. <i>Introduction</i>	25
2.4.2. <i>Metadata Table</i>	25
2.4.3. <i>Metadata Reference Table</i>	27
2.5. EXTENSION MECHANISM	29
2.5.1. <i>Introduction</i>	29
2.5.2. <i>Extensions</i>	30
3. REGISTERED EXTENSIONS	32
3.1. FEATURES	32
3.1.1. <i>GeoPackage Geometry Types Extension</i>	32
3.1.2. <i>User-Defined Geometry Types Extension</i>	32
3.1.3. <i>Rtree Spatial Indexes</i>	33
3.1.4. <i>Geometry Type Triggers</i>	34
3.1.5. <i>SRS_ID Triggers</i>	34
3.2. TILES	35
3.2.1. <i>Zoom Levels</i>	35
3.2.2. <i>Tile Encoding WEBP</i>	36
4. SECURITY CONSIDERATIONS	36
ANNEX A CONFORMANCE / ABSTRACT TEST SUITE (NORMATIVE)	37
A.1 BASE	37
A.2 OPTIONS	42
A.3 REGISTERED EXTENSIONS	69

ANNEX B	BACKGROUND AND CONTEXT (INFORMATIVE)	82
B.1	PREFACE.....	82
B.2	SUBMITTING ORGANIZATIONS	82
B.3	REVISION HISTORY	83
B.4	CHANGES TO THE OGC® ABSTRACT SPECIFICATION	94
B.5	CHANGES TO OGC® IMPLEMENTATION STANDARDS	94
B.6	POTENTIAL FUTURE WORK	94
B.7	CONTRIBUTORS.....	95
B.8	TERMS AND DEFINITIONS.....	96
B.9	CONVENTIONS.....	97
B.10	UML NOTATION.....	99
B.11	GEOPACKAGE TABLES DETAILED DIAGRAM	101
B.12	GEOPACKAGE MINIMAL TABLES FOR FEATURES DIAGRAM	102
B.13	GEOPACKAGE MINIMAL TABLES FOR TILES DIAGRAM.....	103
ANNEX C	TABLE DEFINITION SQL (NORMATIVE)	104
C.1	GPKG_SPATIAL_REF_SYS	104
C.2	GPKG_CONTENTS	104
C.3	GPKG_GEOMETRY_COLUMNS.....	105
C.4	SAMPLE_FEATURE_TABLE (INFORMATIVE)	105
C.5	GPKG_TILE_MATRIX_SET	106
C.6	GPKG_TILE_MATRIX	106
C.7	SAMPLE_TILE_PYRAMID (INFORMATIVE).....	107
C.8	GPKG_DATA_COLUMNS	107
C.9	GPKG_DATA_COLUMN_CONSTRAINTS.....	108
C.10	GPKG_METADATA	108
C.11	GPKG_METADATA_REFERENCE.....	108
C.12	GPKG_EXTENSIONS	109
ANNEX D	TRIGGER DEFINITION SQL (INFORMATIVE)	110
D.1	GPKG_TILE_MATRIX.....	110
D.2	METADATA.....	111
D.3	METADATA_REFERENCE.....	112
D.4	SAMPLE_FEATURE_TABLE	114
D.5	SAMPLE_TILE_PYRAMID.....	115
ANNEX E	GEOMETRY TYPES (NORMATIVE)	117
ANNEX F	TILES ZOOM TIMES TWO EXAMPLE (INFORMATIVE)	118
ANNEX G	HIERARCHICAL METADATA EXAMPLE (INFORMATIVE)	119
ANNEX H	RASTER OR TILE METADATA EXAMPLE (INFORMATIVE)	126
ANNEX I	GEOPACKAGE EXTENSION TEMPLATE (NORMATIVE)	127
ANNEX J	GEOPACKAGE GEOMETRY TYPES EXTENSION TEMPLATE (NORMATIVE)	129
ANNEX K	USER-DEFINED GEOMETRY TYPES EXTENSION TEMPLATE (NORMATIVE)	131
ANNEX L	R TREE SPATIAL INDEX EXTENSION (NORMATIVE)	133
ANNEX M	GEOMETRY TYPE TRIGGERS EXTENSION (NORMATIVE)	137
ANNEX N	GEOMETRY SRS_ID TRIGGERS EXTENSION (NORMATIVE)	139
ANNEX O	ZOOM OTHER INTERVALS EXTENSION (NORMATIVE)	141
ANNEX P	TILE ENCODING WEBP EXTENSION	143

ANNEX Q	NORMATIVE REFERENCES (NORMATIVE)	144
ANNEX R	BIBLIOGRAPHY (INFORMATIVE)	146

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[B1][B2], 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 OGC® Encoding Standard defines GeoPackages 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.

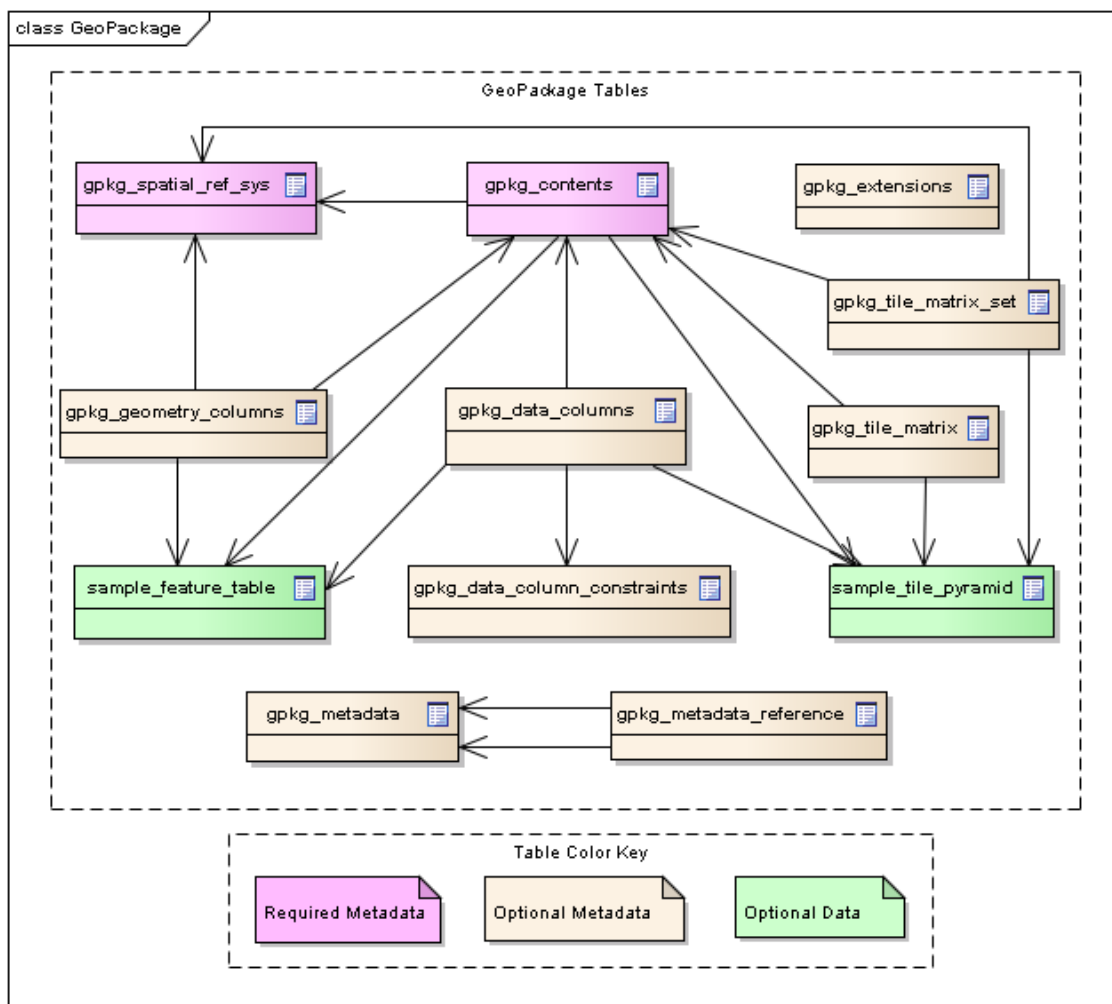
A **GeoPackage** is a platform-independent SQLite [5] database file that contains GeoPackage data and metadata tables shown in Figure 1 below, with specified definitions, integrity assertions, format limitations and content constraints. The allowable content of a **GeoPackage** is entirely defined in this specification.

An **Extended GeoPackage** is a **GeoPackage** that contains any additional data elements (tables or columns) or SQL constructs (data types, functions, indexes, constraints or triggers) that are not specified in this encoding standard.

A **GeoPackage** 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 metadata CAN describe GeoPackage data contents and identify external data synchronization sources and targets. A GeoPackage MAY contain spatial indexes on feature geometries and SQL triggers to maintain indexes and enforce content constraints.

A **GeoPackage SQLite Configuration** consists of the SQLite 3 software library and a set of compile- and runtime configurations options.

A **GeoPackage SQLite Extension** is a SQLite loadable extension that MAY provide SQL functions [12] to support spatial indexes and SQL triggers linked to a SQLite library with specified configuration requirements to provide SQL API [1][2][3][4] access to a GeoPackage. This specification does not address the issues listed in the Potential Future Work clause in Annex B, which MAY be addressed in a subsequent version of this specification or by other specifications.

Figure 1- GeoPackage Tables Overview¹

Attention is drawn to the possibility that some of the elements of this document *MAY* be the subject of patent rights. Open Geospatial Consortium Inc. *SHALL* not be held responsible for identifying any or all such patent rights. However, to date, no such rights have been claimed or identified.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they *MAY* be aware that might be infringed by any implementation of the specification set forth in this document, and to provide supporting documentation.

¹ Also see Figure 4 GeoPackage Tables Details in B.13

OGC® GeoPackage Encoding Standard

1. Base

The required capabilities specified in this clause serve as the base for options specified in clause 2 and extensions specified in clause 3. All `gpkg_*` tables and views and all tiles user data tables specified in this standard SHALL have only the specified columns and table constraints. Any features user data tables MAY have columns in addition to those specified. All specified table and column name values SHALL be lowercase.

1.1. Core

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

1.1.1. SQLite Container

The SQLite software library provides a self-contained, single-file, cross-platform, serverless, transactional, open source RDBMS container. The GeoPackage specification defines a SQL database schema designed for use with the SQLite software library. Using SQLite as the basis for GeoPackage simplifies production, distribution and use of GeoPackages and assists in guaranteeing 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: *A GeoPackage SHALL be a SQLite [5] database file using version 3 of the SQLite file format [6][7]. The first 16 bytes of a GeoPackage SHALL contain “SQLite format 3”¹ in ASCII[B4].²*

Req 2: *A GeoPackage SHALL contain 0x47503130 (“GP10” in ASCII) in the application id field of the SQLite database header to indicate that a GeoPackage version 1.0 file.³*

The maximum size of a GeoPackage 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.

1.1.1.1.2. File Extension Name

Req 3: *A GeoPackage SHALL have the file extension “.gpkg”.*

It is RECOMMENDED that **Extended GeoPackages** use the file extension “.gpkgx”, but this is NOT a GeoPackage requirement.

1.1.1.1.3. File Contents

Req 4: *A GeoPackage SHALL only contain data elements, SQL constructs and GeoPackage extensions with the “gpkg” author name specified in this encoding standard.*

In order to guarantee maximum interoperability between applications, **GeoPackages** SHALL NOT contain data elements (tables or columns), SQL constructs (data types, indexes, constraints or triggers) or extensions that are not specified in this encoding standard. SQLite databases that use constructs from the GeoPackage specification but extend that to contain any of these elements are referred to as **Extended GeoPackages** throughout this specification.

Req 5: *The columns of tables in a GeoPackage SHALL only be declared using one of the data types specified in Table 1.*

¹ SQLite version 4 [B25], 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>)

³ With SQLite versions 3.7.17 and later this value MAY be set with the "PRAGMA application_id=1196437808;" SQL statement, where 1196437808 is the 32-bit integer value of 0x47503130. With earlier versions of SQLite the application id can be set by writing the byte sequence 0x47, 0x50, 0x31, 0x30 at offset 68 in the SQLite database file (see http://www.sqlite.org/fileformat2.html#database_header for details).

Table 1 GeoPackage Data Types

Data Type	Size and Description
BOOLEAN	A boolean value representing true or false. Stored as SQLite INTEGER with value 0 for false or 1 for true
TINYINT	8-bit signed two's complement integer. Stored as SQLite INTEGER with values in the range [-128, 127]
SMALLINT	16-bit signed two's complement integer. Stored as SQLite INTEGER with values in the range [-32768, 32767]
MEDIUMINT	32-bit signed two's complement integer. Stored as SQLite INTEGER with values in the range [-2147483648, 2147483647]
INT, INTEGER	64-bit signed two's complement integer. Stored as SQLite INTEGER with values in the range [-9223372036854775808, 9223372036854775807]
FLOAT	32-bit IEEE floating point number Stored as SQLite REAL limited to values that can be represented as a 4-byte IEEE floating point number
DOUBLE, REAL	64-bit IEEE floating point number Stored as SQLite REAL
TEXT{(maxchar_count)}	Variable length string encoded in either UTF-8 or UTF-16, determined by PRAGMA encoding; see http://www.sqlite.org/pragmas.html#pragma_encoding The optional maxchar_count defines the maximum number of characters in the string. If not specified, the length is unbounded. The count is provided for informational purposes, and applications MAY choose to truncate longer strings if encountered. When present, it is best practice for applications to adhere to the character count. Stored as SQLite TEXT
BLOB{(max_size)}	Variable length binary data. The optional max_size defines the maximum number of bytes in the blob. If not specified, the length is unbounded. The size is provided for informational purposes. When present, it is best practice for applications to adhere to the maximum blob size. Stored as SQLite BLOB
<geometry_type_name>	<geometry_type_name> is one of the geometry types listed in Annex E encoded per clause 2.1.3 or a user-defined geometry type encoded per clause 3.1.2 and Annex K. Geometry Types XY, XYZ, XYM and XYZM geometries use the same data type. Stored as SQLite BLOB
DATE	ISO-8601 date string in the form YYYY-MM-DD encoded in either UTF-8 or UTF-16. See TEXT. Stored as SQLite TEXT
DATETIME	ISO-8601 date/time string in the form YYYY-MM-DDTHH:MM:SS.SSSZ with T separator character and Z suffix for coordinated universal time (UTC) encoded in either UTF-8 or UTF-16. See TEXT. Stored as SQLite TEXT

1.1.1.1.4. File Integrity

Req 6: *The SQLite PRAGMA integrity_check SQL command SHALL return “ok” for a GeoPackage.*

Req 7: *The SQLite PRAGMA foreign_key_check() SQL with no parameter value SHALL return an empty result set indicating no invalid foreign key values for a GeoPackage.*

1.1.1.2. API

1.1.1.2.1. Structured Query Language (SQL)

Req 8: *A GeoPackage SQLite Configuration SHALL provide SQL access to GeoPackage contents via SQLite version 3 [6] software APIs.¹*

1.1.1.2.2. Geopackage SQLite Configuration

The SQLite [8] library has many [compile time](#) and [run time](#) options that MAY be used to configure SQLite for different uses. Certain elements of the GeoPackage specification depend on the availability of SQLite functionality at runtime. This clause specifies the set of compile- and runtime options that SHALL or SHALL NOT be used.

Req 9: *Every GeoPackage SQLite Configuration SHALL have the SQLite library compile and run time options specified in clause 1.1.1.2.2Table 2.*

Table 2 Every GeoPackage SQLite Configuration

Setting	Option	Shall / Not	Discussion
compile	SQLITE_OMIT_*	Not	SHALL NOT include any OMIT options from http://www.sqlite.org/compile.html#omitfeatures
run	PRAGMA foreign_keys	Not (OFF)	Foreign key constraints are used to maintain GeoPackage relational integrity.

1.1.2. Spatial Reference Systems

1.1.2.1. Data

1.1.2.1.1. Table Definition

Req 10: *A GeoPackage SHALL include a gpkg_spatial_ref_sys table per clause 1.1.2.1.1,Table 3 and Table 18.*

A table 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.

¹ New applications should use the latest available SQLite version software [8]

The `gpkg_spatial_ref_sys` table includes at a minimum the columns specified in SQL/MM (ISO 13249-3) [12] and shown in Table 3 below containing data that defines spatial reference systems. Views of this table MAY be used to provide compatibility with the SQL/MM [12] (Table 19) and OGC Simple Features SQL [9] (Table 20) specifications.

Table 3 Spatial Ref Sys Table or View Definition

Column Name	Column Type	Column Description	Null	Key
<code>srs_name</code>	TEXT	Human readable name of this SRS	no	
<code>srs_id</code>	INTEGER	Unique identifier for each Spatial Reference System within a GeoPackage	no	PK
<code>organization</code>	TEXT	Case-insensitive name of the defining organization e.g. EPSG or epsg	no	
<code>organization_coordsys_id</code>	INTEGER	Numeric ID of the Spatial Reference System assigned by the organization	no	
<code>definition</code>	TEXT	Well-known Text Representation of the Spatial Reference System	no	
<code>description</code>	TEXT	Human readable description of this SRS	yes	

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

1.1.2.1.2. Table Data Values

Req 11: *The `gpkg_spatial_ref_sys` table in a GeoPackage SHALL contain a record for organization EPSG or epsg [B3] and organization_coordsys_id 4326 [13][14] for WGS-84 [15], 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 12: *The `gpkg_spatial_ref_sys` table in a GeoPackage 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 13: *A GeoPackage SHALL include a `gpkg_contents` table per clause 1.1.3.1.1, Table 4 and Table 21.*

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 4 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 in an Extended GeoPackage.	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	DATETIME	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)	
min_x	DOUBLE	Bounding box minimum easting or longitude for all content in table_name	yes		
min_y	DOUBLE	Bounding box minimum northing or latitude for all content in table_name	yes		
max_x	DOUBLE	Bounding box maximum easting or longitude for all content in table_name	yes		
max_y	DOUBLE	Bounding box maximum northing or latitude 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; When data_type is tiles, SHALL also match gpkg_tile_matrix_set.srs.id	yes		FK

The gpkg_contents table is intended to provide a list of all geospatial contents in a 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 14: *The table_name column value in a gpkg_contents table row SHALL contain the name of a SQLite table or view.*

Req 15: *Values of the gpkg_contents table last_change column SHALL be in ISO 8601 [29] 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 16: *Values of the gpkg_contents table srs_id column SHALL reference values in the gpkg_spatial_ref_sys table srs_id column.*

2. Options

The optional capabilities specified in this clause depend on the required capabilities specified in clause 1 Base above. Each subclause of this clause defines an indivisible module of functionality that can be used in GeoPackages. These modules are referred to as options. GeoPackages MAY use one or more options defined in this section. GeoPackages MAY omit the tables for options that are not used. As a minimum, a GeoPackage SHALL contain one user data table as defined by the Features or Tiles options in clauses 2.1.6.1.1 and 2.2.8.1.1 respectively.

Req 17: *A GeoPackage SHALL contain features per clause 2.1 and/or tiles per clause 2.2 and row(s) in the gpkg_contents table with lowercase data_type column values of "features" and/or "tiles" describing the user data tables.*

2.1. Features

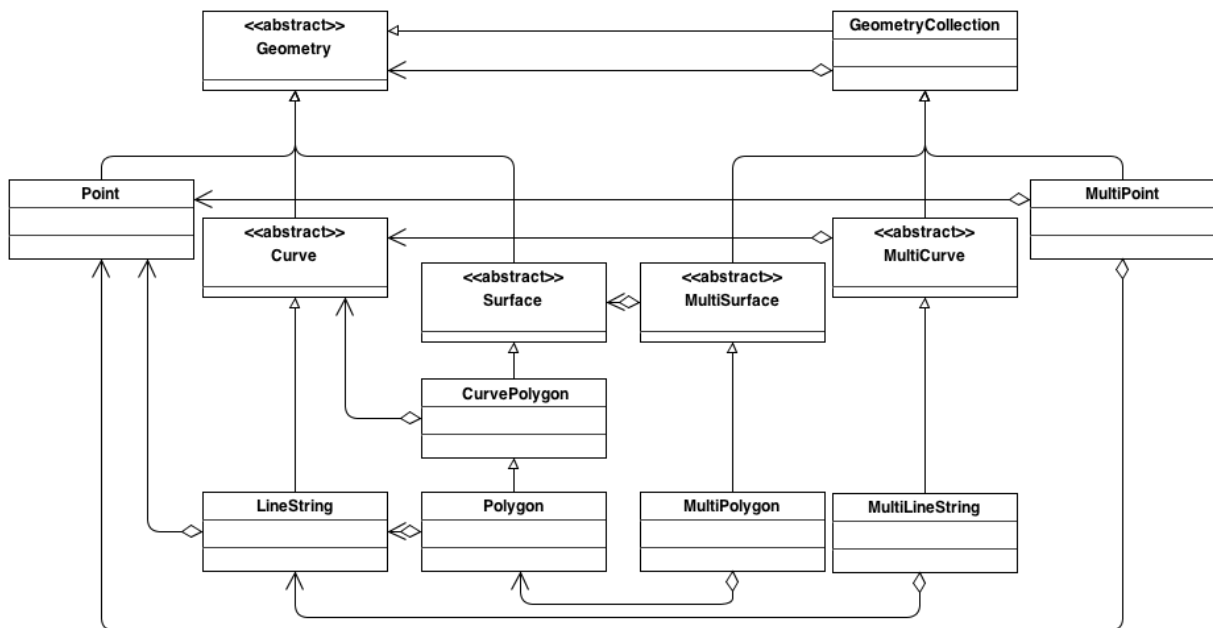
2.1.1. Simple Features SQL Introduction

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 [9][10][11][12] 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, "simple" features are geolocated using a linear geometry subset of the SQL/MM (ISO 13249-3) [12] geometry model shown in figure 2 below.

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

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.

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[12].

- 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 more 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.

2.1.2. Contents

2.1.2.1. Data

2.1.2.1.1. Contents Table – Features Row

Req 18: *The `gpkg_contents` table SHALL contain a row with a lowercase `data_type` column value of “features” for each vector features user data table or view.*

2.1.3. Geometry Encoding

2.1.3.1. Data

2.1.3.1.1. BLOB Format

Req 19: *A `GeoPackage` SHALL store feature table geometries with or without optional elevation (Z) and/or measure (M) values in SQL BLOBs using the `StandardGeoPackageBinary` format specified in Table 5 and clause 2.1.3.1.1.*

Table 5 GeoPackage SQL Geometry Binary Format

```

GeoPackageBinaryHeader {
    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
}

```

bit layout of flags byte:

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

flag bits use:

R: reserved for future use; set to 0

X: GeoPackageBinary type

0: StandardGeoPackageBinary. See below.

1: ExtendedGeoPackageBinary See clause 3.1.2, Annex K.

Y: empty geometry flag

0: non-empty geometry

1: empty geometry

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

code value	description	envelope 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)

```

StandardGeoPackageBinary {
    GeoPackageBinaryHeader header; // The X bit in the header flags field
                                   // must be set to 0.
    WKBGeometry geometry;         // per ISO 13249-3[12] clause 5.1.46 123
}

```

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

² WKB geometry types are restricted to 0, 1 and 2-dimensional geometric objects that exist in 2, 3 or 4-dimensional coordinate space; they are not geographic or geodesic geometry types.

³ The axis order in WKB is always (x,y[,z][,m]) where x is easting or longitude, y is northing or latitude, z is optional elevation and m is optional measure.

Well-Known Binary as defined in ISO 13249-3 [12] does not provide a standardized encoding for an empty point set (i.e., 'Point Empty' in Well-Known Text). In GeoPackages these points SHALL be encoded as a Point where each coordinate value is set to an IEEE-754 quiet NaN value. GeoPackages SHALL use big endian 0x7ff8000000000000 or little endian 0x0000000000000f87f 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.

2.1.4. SQL Geometry Types

2.1.4.1. Data

2.1.4.1.1. Core Types

Req 20: *A GeoPackage SHALL store feature table geometries with the basic simple feature geometry types (Geometry, Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon, GeomCollection) in Annex E Table 42 in the GeoPackageBinary geometry encoding format.*

2.1.5. Geometry Columns

2.1.5.1. Data

2.1.5.1.1. Table Definition

Req 21: *A GeoPackage 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 6 and Table 22.*

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.

Table 6 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 42 or Table 43 in Annex E	
srs_id	INTEGER	Spatial Reference System ID: gpkg_spatial_ref_sys.srs_id	FK
z	TINYINT	0: z values prohibited 1: z values mandatory 2: z values optional	
m	TINYINT	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.

Views of this table or view MAY be used to provide compatibility with the SQL/MM [12] (Table 23) and OGC Simple Features SQL [9][10][11] (Table 24) specifications.

See Annex C Table Definition SQL (Normative) C.3 gpkg_geometry_columns.

2.1.5.1.2. Table Data Values

Req 22: *The gpkg_geometry_columns table or updateable view SHALL contain one row record for each geometry column in each vector feature user data table (clause 2.1.6) in a GeoPackage.*

Req 23: *Values of the gpkg_geometry_columns table_name column SHALL reference values in the gpkg_contents table_name column for rows with a data_type of “features”.*

Req 24: *The column_name column value in a gpkg_geometry_columns row SHALL be the name of a column in the table or view specified by the table_name column value for that row.*

Req 25: *The geometry_type_name value in a gpkg_geometry_columns row SHALL be one of the uppercase geometry type names specified in Annex E.*

Req 26: *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 27: *The z value in a gpkg_geometry_columns table row SHALL be one of 0, 1, or 2.*

Req 28: *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. Features are rows in a feature table.¹

Req 29: A GeoPackage MAY contain tables or updateable views containing vector features. Every such feature table or view in a GeoPackage SHALL have a column with column type INTEGER and 'PRIMARY KEY AUTOINCREMENT' column constraints per Table 7 and Table 25.

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 [B5] values in the gpkg_metadata_reference table as described in clause 2.4.3 below.

Req 30: A feature table SHALL have only one geometry column.

Feature data models [B23] 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 7: EXAMPLE: Sample Feature Table or View Definition

Column Name	Type	Description	Null	Default	Key
id	INTEGER	Autoincrement primary key	no		PK
geometry	GEOMETRY	GeoPackage Geometry	yes		
text_attribute	TEXT	Text attribute of feature	yes		
real_attribute	REAL	Real attribute of feature	yes		
boolean_attribute	BOOLEAN	Boolean attribute of feature	yes		
raster_or_photo	BLOB	Photograph of the area	yes		

See Annex C Table Definition SQL (Normative) C.4 sample_feature_table

2.1.6.1.2. Table Data Values

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

¹ A GeoPackage is not required to contain any feature data tables. Feature data tables in a GeoPackage MAY be empty

The geometry type of a feature geometry column specified in the `gpkg_geometry_columns` table `geometry_type_name` column is an uppercase name from Table 42 or Table 43 in Annex E.

Req 31: *Feature table geometry columns SHALL contain geometries of the type or assignable for the type specified for the column by the `gpkg_geometry_columns` `geometry_type_name` uppercase column value¹.*

Geometry subtypes are assignable as defined in Annex E, 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.

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 32: *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

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 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 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.

¹ GeoPackage applications MAY use SQL triggers or tests in application code to meet Req 32:.

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 “Tile pyramid” refers to the concept of pyramid structure of tiles of different spatial extent and resolution at different zoom levels, and the tile data itself. “Tile matrix” refers to rows and columns of tiles that all have the same spatial extent and resolution at a particular zoom level. “Tile matrix set” refers to the definition of a tile pyramid’s tiling structure.

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

2.2.2. Contents

2.2.2.1. Data

2.2.2.1.1. Contents Table – Tiles Row

Req 33: *The `gpkg_contents` table SHALL contain a row with a lowercase `data_type` column value of “tiles” for each tile pyramid user data table or view.*

2.2.3. Zoom Levels

In a GeoPackage, zoom levels are integers in sequence from 0 to n that identify tile matrix layers in a tile matrix set that contain tiles of decreasing spatial extent and finer spatial resolution. Adjacent zoom levels immediately precede or follow each other and differ by a value of 1. Pixel sizes are real numbers in the terrain units of the spatial reference system of a tile image specifying the dimensions of the real world area represented by one pixel. Pixel sizes MAY vary by a constant factor or by different factors or intervals between some or all adjacent zoom levels in a tile matrix set. In the commonly used "zoom times two" convention, pixel sizes vary by a factor of 2 between all adjacent zoom levels, as shown in the example in Annex F. Other "zoom other intervals" conventions use different factors or irregular intervals with pixel sizes chosen for intuitive cartographic representation of raster data, or to coincide with the original pixel size of commonly used global image products. See WMTS [16] Annex E for additional examples of both conventions.

¹ Images of multiple MIME types MAY be stored in given table. For example, in a tiles table, image/png format tiles COULD be used for transparency where there is no data on the tile edges, and image/jpeg format tiles 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.

2.2.3.1. Data

2.2.3.1.1. Zoom Times Two

Req 34: *In a GeoPackage that contains a tile pyramid user data table that contains tile data, by default¹, zoom level pixel sizes for that table SHALL vary by a factor of 2 between adjacent zoom levels in the tile matrix table.*

2.2.4. Tile Encoding PNG

2.2.4.1. Data

2.2.4.1.1. MIME Type PNG

Req 35: *In a GeoPackage that contains a tile pyramid user data table that contains tile data that is not MIME type image/jpeg [17][18][19], by default SHALL store that tile data in MIME type image/png [20][21].²*

2.2.5. Tile Encoding JPEG

2.2.5.1. Data

2.2.5.1.1. MIME Type JPEG

Req 36: *In a GeoPackage that contains a tile pyramid user data table that contains tile data that is not MIME type image/png [20][21], by default SHALL store that tile data in MIME type image/jpeg[17][18][19],³*

2.2.6. Tile Matrix Set

2.2.6.1. Data

2.2.6.1.1. Table Definition

Req 37: *A GeoPackage that contains a tile pyramid user data table SHALL contain a `gpkg_tile_matrix_set` table or view per clause 2.2.6.1.1, Table 8 and Table 26.*

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

² See Clause 3.2.2 regarding use of the WebP alternative tile MIME type as a registered extension.

³ See Clause 3.2.2 regarding use of the WebP alternative tile MIME type as a registered extension.

Table 8 Tile Matrix Set Table or View Definition

Column Name	Column Type	Column Description	Null	Default	Key
table_name	TEXT	Tile Pyramid User Data Table Name	no		PK, FK
srs_id	INTEGER	Spatial Reference System ID: gpkg_spatial_ref_sys.srs_id	no		FK
min_x	DOUBLE	Bounding box minimum easting or longitude for all content in table_name	no		
min_y	DOUBLE	Bounding box minimum northing or latitude for all content in table_name	no		
max_x	DOUBLE	Bounding box maximum easting or longitude for all content in table_name	no		
max_y	DOUBLE	Bounding box maximum northing or latitude for all content in table_name	no		

The gpkg_tile_matrix_set table or updateable view defines the minimum bounding box (min_x, min_y, max_x, max_y) and spatial reference system (srs_id) for all content in a tile pyramid user data table.

See Annex C Table Definition SQL (Normative) C.5 gpkg_tile_matrix_set

2.2.6.1.2. Table Data Values

The minimum bounding box defined in the gpkg_tile_matrix_set table or view for a tile pyramid user data table SHALL be exact so that the bounding box coordinates for individual tiles in a tile pyramid MAY be calculated based on the column values for the user data table in the gpkg_tile_matrix table or view. For example, because GeoPackages use the upper left tile origin convention defined in clause 2.2.7.1.2 below, the gpkg_tile_matrix_set (min_x, max_y) ordinate is the upper-left corner of tile (0,0) for all zoom levels in a table_name tile pyramid user data table.

Req 38: *Values of the gpkg_tile_matrix_set.table_name column SHALL reference values in the gpkg_contents table_name column for rows with a data_type of “tiles”.*

Req 39: *The gpkg_tile_matrix_set table or view SHALL contain one row record for each tile pyramid user data table.*

Req 40: *Values of the gpkg_tile_matrix_set.srs_id column SHALL reference values in the gpkg_spatial_ref_sys table srs_id column.*

2.2.7. Tile Matrix

2.2.7.1. Data

2.2.7.1.1. Table Definition

Req 41: *A GeoPackage that contains a tile pyramid user data table SHALL contain a gpkg_tile_matrix table or view per clause 2.2.7.1.1, Table 9 and Table 27.*

Table 9 Tile Matrix Table or View Definition

Column Name	Column Type	Column Description	Null	Default	Key
table_name	TEXT	Tile Pyramid User Data Table Name	no		PK, FK
zoom_level	INTEGER	0 <= zoom_level <= max_level for 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 table or updateable view 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 factors of 2, irregular intervals, or regular intervals other than factors of 2.

See Annex C Table Definition SQL (Normative) C.6 gpkg_tile_matrix

2.2.7.1.2. Table Data Values

Req 42: *Values of the gpkg_tile_matrix table_name column SHALL reference values in the gpkg_contents table_name column for rows with a data_type of “tiles”.*

Req 43: *The gpkg_tile_matrix table or view SHALL contain one row record for each zoom level that contains one or more tiles in each tile pyramid user data table or view.*

The gpkg_tile_matrix table or view MAY contain row records for zoom levels in a tile pyramid user data 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 view¹, as specified by WMTS [16]. 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 44: *The zoom_level column value in a gpkg_tile_matrix table row SHALL not be negative.*

Req 45: *The matrix_width column value in a gpkg_tile_matrix table row SHALL be greater than 0.*

Req 46: *The matrix_height column value in a gpkg_tile_matrix table row SHALL be greater than 0.*

Req 47: *The tile_width column value in a gpkg_tile_matrix table row SHALL be greater than 0.*

Req 48: *The tile_height column value in a gpkg_tile_matrix table row SHALL be greater than 0.*

Req 49: *The pixel_x_size column value in a gpkg_tile_matrix table row SHALL be greater than 0.*

Req 50: *The pixel_y_size column value in a gpkg_tile_matrix table row SHALL be greater than 0.*

Req 51: *The pixel_x_size and pixel_y_size column values for zoom_level column values in a gpkg_tile_matrix 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.² 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.³ This does not affect the informative spatial extent stated by the min/max x/y columns values in the gpkg_contents record for the same table_name, the exact spatial extent stated by the min/max x/y columns values in the gpkg_tile_matrix_set record for the same table name, or the tile matrix width and height at that level.⁴

¹ GeoPackage applications MAY query the gpkg_tile_matrix table or the tile pyramid user data table to determine the minimum and maximum zoom levels for a given tile pyramid table.

² GeoPackage applications MAY query a tile pyramid user data table to determine which tiles are available at each zoom level.

³ GeoPackage applications that insert, update, or delete tile pyramid user data table tiles row records are responsible for maintaining the corresponding descriptive contents of the gpkg_tile_matrix table.

⁴ The gpkg_tile_matrix_set table contains coordinates that define a bounding box as the exact stated spatial extent for all tiles in a tile (matrix set) table. If the geographic extent of the image data contained in tiles at a particular zoom level 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.

2.2.8. Tile Pyramid User Data Tables

2.2.8.1. Data

2.2.8.1.1. Table Definition

Req 52: *Each tile matrix set in a GeoPackage SHALL be stored in a different tile pyramid user data table or updateable view with a unique name that SHALL have a column named “id” with column type INTEGER and 'PRIMARY KEY AUTOINCREMENT' column constraints per clause 2.2.8.1.1, Table 10 and Table 30.*

Table 10 Tile Pyramid User Data 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 matrix_width – 1	no	0	UK
tile_row	INTEGER	0 to gpkg_tile_matrix 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 (Normative) C.7 sample_tile_pyramid

2.2.8.1.2. Table Data Values

Each tile pyramid user data table or view¹ MAY contain tile matrices at zero or more zoom levels of different spatial resolution (map scale).

Req 53: *For each distinct table_name from the gpkg_tile_matrix (tm) table, the tile pyramid (tp) user data table zoom_level column value in a GeoPackage SHALL be in the range min(tm.zoom_level) <= tp.zoom_level <= max(tm.zoom_level).*

Req 54: *For each distinct table_name from the gpkg_tile_matrix(tm) table, the tile pyramid (tp) user data table tile_column column value in a GeoPackage SHALL be in the range 0 <= tp.tile_column <= tm.matrix_width – 1 where the tm and tp zoom_level column values are equal.*

Req 55: *For each distinct table_name from the gpkg_tile_matrix(tm) table, the tile pyramid (tp) user data table tile_row column value in a GeoPackage SHALL be in the range 0 <= tp.tile_row <= tm.matrix_height – 1 where the tm and tp zoom_level column values are equal.*

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

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

2.3. Schema

2.3.1. Schema Introduction

The schema option provides a means to describe the columns of tables in a GeoPackage with more detail than can be captured by the SQL table definition directly. The information provided by this option can be used by applications to, for instance, present data contained in a GeoPackage in a more user-friendly fashion or implement data validation logic.

2.3.2. Data Columns

2.3.2.1. Data

2.3.2.1.1. Table Definition

Req 56: *A GeoPackage MAY contain a table or updateable view named `gpkg_data_columns`. If present it SHALL be defined per clause 2.3.2.1.1, Table 11 and Table 31.*

Table 11 Data Columns Table or View Definition

Column Name	Column Type	Column Description	Null	Key
<code>table_name</code>	TEXT	Name of the tiles or feature table	no	PK
<code>column_name</code>	TEXT	Name of the table column	no	PK
<code>name</code>	TEXT	A human-readable identifier (e.g. short name) for the <code>column_name</code> content	yes	
<code>title</code>	TEXT	A human-readable formal title for the <code>column_name</code> content	yes	
<code>description</code>	TEXT	A human-readable description for the <code>table_name</code> content	yes	
<code>mime_type</code>	TEXT	MIME [21] type of <code>column_name</code> if BLOB type, or NULL for other types	yes	
<code>constraint_name</code>	TEXT	Case sensitive column value constraint name specified by reference to <code>gpkg_data_column_constraints.constraint_name</code>	yes	

GeoPackage applications MAY² use the `gpkg_data_columns` table to store minimal application schema identifying, descriptive and MIME [21] type¹ information about columns

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.

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.

See Annex Annex C Table Definition SQL (Normative) C.8 `gpkg_data_columns`

2.3.2.1.2. Table Data Values

Req 57: *Values of the `gpkg_data_columns` table `table_name` column value SHALL reference values in the `gpkg_contents` table `table_name` column.*

Req 58: *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.*

Req 59: *The `constraint_name` column value in a `gpkg_data_columns` table MAY be NULL. If it is not NULL, it SHALL contain a case sensitive `constraint_name` column value from the `gpkg_data_column_constraints` table.*

2.3.3. Data Column Constraints

2.3.3.1. Data

2.3.3.1.1. Table Definition

Req 60: *A GeoPackage MAY contain a table or updateable view named `gpkg_data_column_constraints`. If present it SHALL be defined per clause 2.3.3.1.1, Table 12 and Table 32.*

The `gpkg_data_column_constraints` table contains data to specify restrictions on basic data type column values. The `constraint_name` column is referenced by `constraint_name` column in the `gpkg_data_columns` table defined in clause 2.3.2.1.1.

¹ GeoPackages MAY contain MIME types other than the raster image types specified in clauses 2.2.4, 2.2.5, and 3.2.2 as feature attributes, but they are not required to do so

Table 12 Data Column Constraints Table or View Definition ¹

Column Name	Column Type	Column Description	Null	Key
constraint_name	TEXT	Case sensitive name of constraint	no	Unique
constraint_type	TEXT	Lowercase type name of constraint: 'range' 'enum' 'glob'	no	Unique
value	TEXT	Specified case sensitive value for 'enum' or 'glob' or NULL for 'range' constraint_type	yes	Unique
min	NUMERIC	Minimum value for 'range' or NULL for 'enum' or 'glob' constraint_type	yes	
minIsInclusive	BOOLEAN	0 (false) if min value is exclusive, or 1 (true) if min value is inclusive	yes	
max	NUMERIC	Maximum value for 'range' or NULL for 'enum' or 'glob' constraint_type	yes	
maxIsInclusive	BOOLEAN	0 (false) if max value is exclusive, or 1 (true) if max value is inclusive	yes	
description	TEXT	For ranges and globs, describes the constraint; for enums, describes the enum value.	yes	

See Annex C Table Definition SQL (Normative) C.9 gpkg_data_column_constraints.

2.3.3.1.2. Table Data Values

The lowercase gpkg_data_column_constraints constraint_type column value specifies the type of constraint: "range", "enum", or "glob" (text pattern match). The case sensitive value column contains an enumerated legal value for constraint_type "enum", a pattern match string for constraint_type "glob", or NULL for constraint_type "range". The set of value column values in rows of constraint_type "enum" with the same constraint_name contains all possible enumerated values for the constraint name. The min and max column values specify the minimum and maximum valid values for constraint_type "range", or are NULL for constraint_type "enum" or "glob". The minIsInclusive and maxIsInclusive column values contain 1 if the min and max values are inclusive, 0 if they are exclusive, or are NULL for constraint_type "enum" or "glob". These restrictions MAY be enforced by SQL triggers or by code in applications that update GeoPackage data values.

¹ The min and max columns are defined as NUMERIC to be able to contain range values for any numeric data column defined with a data type from Table 1. These are the only exceptions to the data type rule stated in Req 5.

Table 13 Sample Data Column Constraints

constraint_name	constraint_type	value	min	minIsInclusive	max	maxIsInclusive
sampleRange	range	NULL	1	true	10	true
sampleEnum	enum	1	NULL	NULL	NULL	NULL
sampleEnum	enum	3	NULL	NULL	NULL	NULL
sampleEnum	enum	5	NULL	NULL	NULL	NULL
sampleEnum	enum	7	NULL	NULL	NULL	NULL
sampleEnum	enum	9	NULL	NULL	NULL	NULL
sampleGlob	glob	[1-2][0-9][0-9][0-9]	NULL	NULL	NULL	NULL

Req 61: *The gpkg_data_column_constraints table MAY be empty. If it contains data, the lowercase constraint_type column values SHALL be one of "range", "enum", or "glob".*

Req 62: *gpkg_data_column_constraint constraint_name values for rows with constraint_type values of 'range' and 'glob' SHALL be unique.*

Req 63: *The gpkg_data_column_constraints table MAY be empty. If it contains rows with constraint_type column values of "range", the value column values for those rows SHALL be NULL.*

Req 64: *The gpkg_data_column_constraints table MAY be empty. If it contains rows with constraint_type column values of "range", the min column values for those rows SHALL be NOT NULL and less than the max column value which shall be NOT NULL.*

Req 65: *The gpkg_data_column_constraints table MAY be empty. If it contains rows with constraint_type column values of "range", the minIsInclusive and maxIsInclusive column values for those rows SHALL be 0 or 1.*

Req 66: *The gpkg_data_column_constraints table MAY be empty. If it contains rows with constraint_type column values of "enum" or "glob", the min, max, minIsInclusive and maxIsInclusive column values for those rows SHALL be NULL.*

Req 67: *The gpkg_data_column_constraints table MAY be empty. If it contains rows with constraint_type column values of "enum" or "glob", the value column SHALL NOT be NULL.*

2.4. Metadata

2.4.1. Introduction

Two tables in a GeoPackage provide a means of storing metadata in MIME [21] 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 [28], Annex B B.5.25 MD_ScopeCode, 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 fashion¹ with more than one level, only that if it is, these tables SHALL be used. Such metadata² and data that relates it to GeoPackage contents SHALL NOT be stored in other tables.

2.4.2. Metadata Table

2.4.2.1. Data

2.4.2.1.1. Table Definition

Req 68: *A GeoPackage MAY contain a table named `gpkg_metadata`. If present it SHALL be defined per clause 2.4.2.1.1, Table 14 and Table 33.*

The first component of GeoPackage metadata is the `gpkg_metadata` table that MAY contain metadata in MIME [21] encodings structured in accordance with any authoritative metadata specification, such as ISO 19115 [28], ISO 19115-2 [B6], ISO 19139 [B7], Dublin Core [B8], CSDGM [B10], DDMS [B12], NMF/NMIS [B13], etc. The GeoPackage interpretation of what constitutes “metadata” is a broad one that includes UML models [B14] encoded in XMI [B15], GML Application Schemas [30], ISO 19110 feature catalogues [B18], OWL [B20] and SKOS taxonomies [B21], etc.

¹: Informative examples of hierarchical metadata are provided in Annex G.

² An informative example of raster image metadata is provided in Annex H

Table 14 Metadata Table Definition

Column Name	Column Type	Column Description	Null	Default	Key
id	INTEGER	Metadata primary key	no		PK
md_scope	TEXT	Case sensitive name of the data scope to which this metadata applies; see table 14 below	no	'dataset'	
md_standard_uri	TEXT	URI [23] reference to the metadata structure definition authority ¹	no		
mime_type	TEXT	MIME [21] encoding of metadata	no	text/xml [24]	
metadata	TEXT	metadata	no	''	

The md_standard_uri data value provides an identifier for the metadata structure (schema) specified by its definition authority. The structure (schema) information could be in whatever encoding is used by the definition authority, e.g. UML [B14], or IDEF1x [B16], or XML/Schema [25][26][27], or RDF/S [B19].

See Annex C Table Definition SQL (Normative) C.10 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 15 below. The initial contents of this table were obtained from the ISO 19115 [40], Annex B B.5.25 MD_ScopeCode code list, which was extended² for use in the GeoPackage specification by addition of entries with “NA” as the scope code column in Table 14.

¹ For example, for ISO 19139 metadata the URI value should be the metadata schema namespace <http://www.isotc211.org/2005/gmd>.

² The scope codes in Table 15 include a very wide set of descriptive information types as “metadata” to describe data.

Table 15 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 ¹
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)
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 ²
schema	NA	Metadata applies to an application schema ³
taxonomy	NA	Metadata applies to a taxonomy or knowledge system ⁴
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 69: *Each md_scope column value in a gpkg_metadata table SHALL be one of the case-sensitive name column values from Table 15 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 70: *A GeoPackage that contains a gpkg_metadata table SHALL contain a gpkg_metadata_reference table per clause 2.4.3.1.1, Table 16 and Table 34.*

¹ ISO 19139 format metadata [B32] is recommended for general-purpose description of geospatial data at the series and dataset metadata scopes.

² 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.

³ 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.

⁴ 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.

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 and 2.2.8. The `gpkg_metadata_reference` table is not required to contain any rows.

Table 16 Metadata Reference Table Definition

Column Name	Col Type	Column Description	Null	Default	Key
reference_scope	TEXT	Lowercase 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 rowid of a row record in the table_name table for reference_scope of 'row' or 'row/col'	yes		
timestamp	DATETIME	timestamp value in ISO 8601 format 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 hierarchy².

See Annex C Table Definition SQL (Normative) C.11 `gpkg_metadata_reference`.

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.

2.4.3.1.2. Table Data Values

Req 71: Every *gpkg_metadata_reference* table *reference_scope* column value SHALL be one of 'geopackage', 'table', 'column', 'row', 'row/col' in lowercase.

Req 72: 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 *gpkg_metadata_reference* table row SHALL have a *table_name* column value that references a value in the *gpkg_contents* *table_name* column.

Req 73: 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 74: 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 75: Every *gpkg_metadata_reference* table row *timestamp* column value SHALL be in ISO 8601 [29] 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 76: Every *gpkg_metadata_reference* table row *md_file_id* column value SHALL be an id column value from the *gpkg_metadata* table.

Req 77: 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. Extension Mechanism

2.5.1. Introduction

A GeoPackage extension is a set of one or more requirements clauses that are documented by filling out the GeoPackage Extension Template in Annex I. A GeoPackage Extension either profiles / extends existing requirements clauses in the geopackage specification or adds new requirements clauses. Existing requirement clause extension examples include additional geometry types, additional SQL geometry functions, and additional tile image formats. New

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

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

requirement clause extension examples include spatial indexes, triggers, additional tables, other BLOB column encodings, and other SQL functions.

GeoPackage extensions are identified by a name of the form <author>_<extension name> where <author> indicates the person or organization that developed and maintains the extension. The author value “gpkg” is reserved for GeoPackage extensions that are developed and maintained by OGC and used in Geopackages. Implementers use their own author names to register other extensions¹ used in Extended GeoPackages.

2.5.2. Extensions

2.5.2.1. Data

2.5.2.1.1. Table Definition

Req 78: A GeoPackage MAY contain a table or updateable view named *gpkg_extensions*. If present this table SHALL be defined per clause 2.5.2.1.1, Table 17 and Table 36

The *gpkg_extensions* table or updateable view in a GeoPackage is used to indicate that a particular extension applies to a GeoPackage, a table in a GeoPackage or a column of a table in a GeoPackage. An application that access a GeoPackage can query the *gpkg_extensions* table instead of the contents of all the user data tables to determine if it has the required capabilities to read or write to tables with extensions, and to “fail fast” and return an error message if it does not.

Table 17 GeoPackage Extensions Table or View Definition

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 case sensitive name of the extension that is required, in the form <author>_<extension name>.	no	Unique
definition	TEXT	Definition of the extension in the form specified by the template in Annex I or reference thereto.	no	
scope	TEXT	Indicates scope of extension effects on readers / writers: "read-write" or "write-only" in lowercase.	no	

See Annex C Table Definition SQL (Normative) C.12 *gpkg_extensions*.

¹ See Req 82:.

2.5.2.1.2. Table Data Values

Req 79: *Every extension of a GeoPackage SHALL be registered in a corresponding row in the `gpkg_extensions` table. The absence of a `gpkg_extensions` table or the absence of rows in `gpkg_extensions` table SHALL both indicate the absence of extensions to a GeoPackage.*

Req 80: *Values of the `gpkg_extensions` `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 81: *The `column_name` column value in a `gpkg_extensions` row SHALL be the name of a column in the table specified by the `table_name` column value for that row, or be NULL.*

Req 82: *Each `extension_name` column value in a `gpkg_extensions` row SHALL be a unique case sensitive 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 defined in this encoding standard or in an OGC Best Practices Document that extends it.*

Complete examples of how to fill out the GeoPackage Extension Template in Annex I are provided by Annex L through Annex P. The definition column value in a `gpkg_extensions` row for those extensions SHALL contain the Annex name as a reference.

Partial examples of how to fill out the GeoPackage Extension Template in Annex I are provided by the templates in Annex J and Annex K. Extension definitions created using those templates and other extension definitions created using Annex I MAY be provided in the definition column, preferably as ASCII text, or as a reference such as a URI [23] or email address whereby the definition may be obtained.

Req 83: *The definition column value in a `gpkg_extensions` row SHALL contain or reference the text that results from documenting an extension by filling out the GeoPackage Extension Template in Annex I.*

Some extensions do not impose any additional requirements on software that accesses a GeoPackage in a read-only fashion. An example of this is an extension that defines an SQL trigger that uses a non-standard SQL function defined in a GeoPackage SQLite Extension. Triggers are only invoked when data is written to the GeoPackage, so usage of this type of extension can be safely ignored for read-only access. This is indicated by a `gpkg_extensions.scope` column value of “write_only”.

Req 84: *The `scope` column value in a `gpkg_extensions` row SHALL be lowercase “read-write” for an extension that affects both readers and writers, or “write-only” for an extension that affects only writers.*

3. Registered Extensions

This clause specifies requirements for GeoPackage extensions. Definitions of those extensions in the form specified by the template in Annex I are provided in Annexes J through P.

3.1. Features

The extensions defined in the following sub clauses MAY be implemented in a GeoPackage that implements features per clause 2.1.

3.1.1. GeoPackage Geometry Types Extension

3.1.1.1. Data

This extension of clause 2.1.4 partially described in Annex J defines additional GeoPackage geometry types.

3.1.1.1.1. Extension Types

Req 85: *A GeoPackage MAY store feature table geometries with the extended non-linear geometry types (CircularString, CompoundCurve, CurvePolygon, MultiCurve, MultiSurface, Curve, Surface) in Annex E. If it does so, they SHALL be encoded in the GeoPackageBinary geometry format.*

3.1.1.1.2. Extensions Types - Extensions Name

Req 86: *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 uppercase name of the extension geometry type from Annex E used in a GeoPackage.*

3.1.1.1.3. Extensions Types - Extensions Row

Req 87: *A GeoPackage that contains a gpkg_geometry_columns table or updateable view with row records that specify extension geometry_type_name column values SHALL contain a gpkg_extensions table that contains row records with table_name and column_name values from the gpkg_geometry_columns row records that identify extension type uses, and extension_name column values for each of those geometry types constructed per clause 3.1.1.1.2.*

3.1.2. User-Defined Geometry Types Extension

3.1.2.1. Data

This extension of clauses 2.1.3, 2.1.4, 2.1.5 and 3.1.1 partially described in Annex K enables encoding of additional user-defined geometry types in ExtendedGeoPackageBinary format in an Extended GeoPackage.

3.1.2.1.1. Extensions Encoding

Req 88: *The ExtendedGeoPackageBinary format defined in Annex K SHALL be used to encode geometry types other than those specified in clauses 2.1.4 and 3.1.1 and listed in Annex E.*

3.1.2.1.2. Extensions Encoding - Extensions Name

Req 89: *An extension name to specify a feature geometry extension type encoded in the ExtendedGeoPackageBinary format SHALL be defined for an author name that is NOT “gpkg” using the “<author_name>_geom_<gname>” template where <gname> is the uppercase name of an extension geometry type NOT listed in Annex E used in a GeoPackage.*

3.1.2.1.3. Extensions Encoding- Extensions Row

Req 90: *An Extended GeoPackage that contains a gpkg_geometry_columns table or updateable view with row records that specify extension geometry_type_name column values other than those specified in clauses 2.1.4 and 3.1.1 and listed in Annex E SHALL contain a gpkg_extensions table that contains row records with table_name and column_name values from the gpkg_geometry_columns row records that identify extension type uses, and extension_name column values for each of those geometry type constructed per clause 3.1.2.1.2.*

3.1.2.1.4. Geometry Columns Geometry Type Name

Req 91: *The geometry_type_name value in a gpkg_geometry_columns row SHALL be the Req 89: extension name in uppercase.*

3.1.3. Rtree Spatial Indexes

3.1.3.1. Data

This extension described in Annex L adds a new capability for spatially indexing columns with geometries encoded per clause 2.1.3 and 3.1.2.

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

Req 92: *A GeoPackage 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 L.*

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

3.1.3.1.2. Spatial Indexes - Extensions Name

Req 93: *The “gpkg_rtree_index” extension name SHALL be used as a gpkg_extensions 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 94: *A GeoPackage 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

This extension described in Annex M adds a new geometry type triggers capability for columns with geometries encoded per clause 2.1.3 and 3.1.2.

3.1.4.1.1. Geometry Type Triggers – Implementation

Req 95: *A GeoPackage SHALL include the SQL insert and update triggers specified in Annex M on every geometry column to enforce the geometry type values specified for those columns in the gpkg_geometry_columns table.*

3.1.4.1.2. Geometry Type Triggers – Extensions Name

Req 96: *The “gpkg_geometry_type_trigger” extension name SHALL be used as a gpkg_extensions table extension name column value to specify implementation of geometry type triggers.*

3.1.4.1.3. Geometry Type Triggers – Extensions Row

Req 97: *A GeoPackage that implements geometry type triggers on 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

This extension described in Annex N adds a new srs_id triggers capability for columns with geometries encoded per clause 2.1.3 and 3.1.2.

3.1.5.1.1. SRS_ID Triggers – Implementation

Req 98: *A GeoPackage SHALL include the SQL insert and update triggers specified in Annex Non every geometry column to enforce the srs_id values specified for those columns in the gpkg_geometry_columns table.*

3.1.5.1.2. SRS_ID Triggers – Extensions Name

Req 99: *The “gpkg_srs_id_trigger” extension name SHALL be used as a gpkg_extensions table extension name column value to specify implementation of SRS_ID triggers specified in Annex N.*

3.1.5.1.3. SRS_ID Triggers – Extensions Row

Req 100: *A GeoPackage 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

The extensions defined in the following sub clauses MAY be implemented in a GeoPackage that implements tiles per clause 2.2.

3.2.1. Zoom Levels

3.2.1.1. Data

This extension of clause 2.2.3 described in Annex O allows zoom level intervals other than a factor of two.

3.2.1.1.1. Zoom Other Intervals

Tile pyramid user data tables MAY have pixel sizes that vary by irregular intervals or by regular intervals other than a factor of two (the default) between adjacent zoom levels. Zoom Other – Extensions Name

Req 101: *The “gpkg_zoom_other” extension name SHALL be used as a gpkg_extensions table extension name column value to specify implementation of other zoom intervals on a tile pyramid user data table as specified in Annex O.*

3.2.1.1.2. Zoom Other – Extensions Row

Req 102: *A GeoPackage that implements other zoom intervals SHALL have a gpkg_extensions table that contains a row for each tile pyramid 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

This extension of clauses 2.2.4 and 2.2.5 described in Annex P allows encoding of tile images in WebP format.

3.2.2.2. WEBP MIME Type

A GeoPackage that contains a tile pyramid user data table that contains tile data MAY store tile_data in MIME type image/x-webp[22].

3.2.2.2.1. WEBP -- Extensions Name

Req 103: *The “gpkg_webp” extension name SHALL be used as a gpkg_extensions table extension name column value to specify storage of tile pyramid images in WEBP format as specified in Annex P.*

3.2.2.2.2. WEBP -- Extensions Row

Req 104: *A GeoPackage that contains tile pyramid user data tables with tile_data columns that contain 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”.*

4. 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

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 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

Test Case ID: /base/core/container/data/file_format/application_id

Test Purpose: Verify that the SQLite database header application id field indicates GeoPackage version 1.0

Test Method: Pass if the application id field of the SQLite database header contains “GP10” in ASCII.

Reference: Clause 1.1.1.1.1 Req 2:

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 extension is ".gpkg"

Test Method: Pass if the geopackage file extension is ".gpkg"

Reference: Clause 1.1.1.1.2 Req 3:

Test Type: Basic

A.1.1.1.1.3 File Contents

Test Case ID: /base/core/container/data/file_contents

Test Purpose: Verify that the Geopackage only contains specified contents

Test Method:

- 1) For each gpkg_* table_name
 - a) PRAGMA table_info(table_name)
 - b) Continue if returns an empty result set
 - c) Fail if column definitions returned by PRAGMA table_info do not match column definitions for the table in Annex C.
- 2) Do test /opt/features/vector_features/data/feature_table_integer_primary_key
- 3) Do test /opt/features/vector/features/data/feature_table_one_geometry_column
- 4) Do test /opt/tiles/contents/data/tiles_row
- 5) SELECT extension_name FROM gpkg_contents
- 6) For each row from #4
 - a) Fail if the substring before the first “_” is not “gpkg”

7) Pass if no fails

Reference: Clause 1.1.1.1.3 Req 4:

Test Type: Basic

Test Case ID: /base/core/container/data/table_data_types

Test Purpose: Verify that the data types of GeoPackage columns include only the types specified by Clause 1.1.1.1.3 Table 1

Test Method:

- 1) SELECT table_name FROM gpkg_contents WHERE data_type = 'features'
- 2) Not testable if returns empty set
- 3) For each row table name from step 1
 - a) PRAGMA table_info(table_name)
 - b) Fail if returns empty set
 - c) For each row type column value
 - i. Fail if value is not one of the data type names specified by Clause 1.1.1.1.3 Table 1
- 4) Pass if no fails

Reference: Clause 1.1.1.1.3 Table 1 Req 5:

Test Type: Basic

A.1.1.1.1.4 Integrity Check

Test Case ID: /base/core/container/data/file_integrity

Test Purpose: Verify that the geopackage passes the SQLite integrity check.

Test Method: Pass if PRAGMA integrity_check returns "ok"

Reference: Clause 1.1.1.1.4 Req 6:

Test Type: Capability

Test Case ID: /base/core/container/data/foreign_key_integrity

Test Purpose: Verify that the geopackage passes the SQLite foreign_key_check.

Test Method: Pass if PRAGMA foreign_key_check() with no parameter value returns an empty result set

Reference: Clause 1.1.1.1.4 Req 7:

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) sqlite3_exec('SELECT * FROM sqlite_master;')

- 2) Fail if returns an SQL error.
- 3) Pass otherwise

Reference: Clause 1.1.1.2.1 Req 8:

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) For each “SQLITE_OMIT_*” <option> listed at <http://www.sqlite.org/compile.html#omitfeatures>
 - a) SELECT sqlite_compileoption_used('SQLITE_OMIT_<option>')
 - b) Fail if returns 1
- 2) PRAGMA foreign_keys
- 3) Fail if returns 0
- 4) Pass otherwise

Reference: Clause 1.1.1.2.2 Req 9:

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 gpkg_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, and primary key constraints match all of those in the contents of C.1Table 18. 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 10:

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", 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 11:

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 1.1.2.1.2 Req 12:

Test Type: Capability

A.1.1.3 Contents

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.2Table 19. Column

order, check constraint and trigger definitions, and other column definitions in the returned sql are irrelevant.

4) Fail Otherwise

Reference: Clause 1.1.3.1.1 Req 13:

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:

- 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 14:

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 [29]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 15:

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 reference gpkg_spatial_ref_sys srs_id column values.

Test Method:

- 1) PRAGMA foreign_key_check('gpkg_contents')
- 2) Fail if does not return an empty result set

Reference: Clause 1.1.3.1.2 Req 16:

Test Type: Capability

A.2 Options

Test Case ID: /opt/valid_geopackage

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 17:

Test Type: Capability

A.2.1 Features

A.2.1.1 Simple Features SQL Introduction

A.2.1.2 Contents

A.2.1.2.1 Data

A.2.1.2.1.1 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 a feature table for every row with a data_type column value of “features”

Test Method:

- 1) Execute test /opt/features/vector_features/data/feature_table_integer_primary_key

Reference: Clause 2.1.2.1.1 Req 18:

Test Type: Capability

A.2.1.3 Geometry Encoding

A.2.1.3.1 Data

A.2.1.3.1.1 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 StandardGeoPackageBinary 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
 - a. SELECT cn FROM tn
 - b. Not testable if none found
 - c. For each cn value from step a
 - i. Fail if the first two bytes of each gc are not "GP"
 - ii. Fail if gc.version_number is not 0
 - iii. Fail if gc.flags.GeopackageBinary type != 0
 - iv. 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 19:

Test Type: Capability

A.2.1.4 SQL Geometry Types

A.2.1.4.1 Data

A.2.1.4.1.1 Core Types

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 1 of cn from #1 are a geometry type value from Annex E Table 42, then
 - a. Log cn.header values, wkb endianness and geometry type
 - b. If cn.wkb is not correctly encoded per ISO 13249-3 clause 5.1.46 then log fail
 - c. If cn.flags.E is 1 - 4 and some cn.wkbx is outside of cn.envelope.minx,maxx then log fail
 - d. If cn.flags.E is 1 - 4 and some gc.wkby is outside of cn.envelope.miny,maxy then log fail
 - e. If cn.flags.E is 2,4 and some gc.wkb.z is outside of cnenvelope.minz,maxz then log fail
 - f. If cn.flags.E is 3,4 and some gc.wkb.m is outside of cn.envelope.minm,maxm then log fail
 - g. If cn.flags.E is 5-7 then log fail

- h. Otherwise log pass
- 6) Pass if log contains pass and no fails

Reference: Clause 2.1.4.1.1 Req 20:

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 E, 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 GP headers containing big and little endian WKBs for 0-1 envelope contents indicator codes for every geometry type value from Annex E Table 42 without and with z and/or m values.
- 4) Fail otherwise

Reference: Clause 2.1.4.1.1 Req 20:

Test Type: Capability

A.2.1.5 Geometry Columns

A.2.1.5.1 Data

A.2.1.5.1.1 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 21. 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 21:

Test Type: Basic

A.2.1.5.1.2 Table Data Values

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

Test Purpose: Verify that gpkg_geometry_columns contains one row record for each geometry column in each vector feature user data table.

Test Method:

- 1) SELECT table_name FROM gpkg_contents WHERE data_type = 'features'
- 2) Not testable if returns an empty result set
- 3) SELECT table_name FROM gpkg_contents WHERE data_type = 'features' AND table_name NOT IN (SELECT table_name FROM gpkg_geometry_columns)
- 4) Fail if result set is not empty

Reference: Clause 2.1.5.1.2 Req 22:

Test Type: Capability

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) PRAGMA foreign_key_check('geometry_columns')
- 2) Fail if returns any rows with a fourth column foreign key index value of 1 (gpkg_contents)

Reference: Clause 2.1.5.1.2 Req 23:

Test Type: Capability

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

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.
- 4) Pass if no fails.

Reference: Clause 2.1.5.1.2 Req 24:

Test Type: Capability

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

Test Purpose: Verify 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_name value is not in Table 42 or Table 43 in Annex E
- 4) Pass if no fails.

Reference: Clause 2.1.5.1.2 Req 25:

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 valid.

Test Method:

- 1) PRAGMA foreign_key_check('gpkg_geometry_columns')
- 2) Fail if returns any rows with a fourth column foreign key index value of 0

Reference: Clause 2.1.5.1.2 Req 26:

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:

- 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 (0,1,2)
- 4) Fail if does not return an empty result set
- 5) Pass otherwise.

Reference: Clause 2.1.5.1.2 Req 27:

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 (0,1,2)
- 4) Fail if does not return an empty result set
- 5) Pass otherwise.

Reference: Clause 2.1.5.1.2 Req 28:

Test Type: Capability

A.2.1.6 Vector Features User Data Tables

A.2.1.6.1 Data**A.2.1.6.1.1 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 WHERE 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 29:**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 WHERE data_type = 'features'
- 2) Not testable if returns an empty result set
- 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 30:**Test Type:** Capability**A.2.1.6.1.2 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 31:

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 32:

Test Type: Capability

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) Not testable 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 33:

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 factors 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 gpkg_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 gpkg_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 gpkg_tile_matrix
 WHERE table_name = selected table name ORDER BY zoom_level ASC
 - 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 factors of two
- 4) Pass if no fails

Reference: Clause 2.2.3.1.1 Req 34:

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 gpkg_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 gpkg_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 35:

**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

```

WHEN (SELECT tbl_name FROM sqlite_master WHERE tbl_name =
'gpkg_extensions') = 'gpkg_extensions' THEN
(SELECT table_name FROM gpkg_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 gpkg_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 36:

**Test Type:** Capability

**A.2.2.5 Tile Matrix Set****A.2.2.5.1 Data****A.2.2.5.1.1 Table Definition****Test Case ID:** /opt/tiles/gpkg\_tile\_matrix\_set /data/table\_def**Test Purpose:** Verify that the gpkg\_tile\_matrix\_set table exists and has the correct definition.**Test Method:**

- 1) SELECT sql FROM sqlite\_master WHERE type = 'table' AND tbl\_name = 'gpkg\_tile\_matrix\_set'
- 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.5Annex C Table 26. 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 37:**Test Type:** Capability**A.2.2.5.1.2 Table Data Values****Test Case ID:** /opt/tiles/gpkg\_tile\_matrix\_set/data\_values\_table\_name**Test Purpose:** Verify that values of the gpkg\_tile\_matrix\_set table\_name column reference values in the gpkg\_contents table\_name column for rows with a data type of “tiles”.**Test Method:**

- 1) SELECT table\_name FROM gpkg\_tile\_matrix\_set
- 2) Not testable if returns an empty result set
- 3) SELECT table\_name FROM gpkg\_tile\_matrix\_set tms WHERE table\_name NOT IN (SELECT table\_name FROM gpkg\_contents gc WHERE tms.table\_name = gc.table\_name AND gc.data\_type != 'tiles')
- 4) Fail if result set contains any rows
- 5) Pass otherwise

**Reference:** Clause 2.2.6.1.2 Req 38:**Test Type:** Capability**Test Case ID:** /opt/tiles/gpkg\_tile\_matrix\_set/data/data\_values\_row\_record**Test Purpose:** Verify that the gpkg\_tile\_matrix\_set table contains a row record for each tile pyramid 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 sql FROM sqlite\_master WHERE type='table' AND tbl\_name = '<user\_data\_tiles\_table>'
  - b. Fail if returns an empty result set
- 4) Pass if no fails

**Reference:** Clause 2.2.6.1.2 Req 39:

**Test Type:** Capability

**Test Case ID:** /opt/tiles/gpkg\_tile\_matrix\_set/data/data\_values\_srs\_id

**Test Purpose:** Verify that the gpkg\_tile\_matrix\_set table srs\_id column values reference gpkg\_spatial\_ref\_sys srs\_id column values.

**Test Method:**

- 1) PRAGMA foreign\_key\_check('gpkg\_geometry\_columns')
- 2) Fail if returns any rows with a fourth column foreign key index value of 1 (gpkg\_spatial\_ref\_sys)

**Reference:** Clause 2.2.6.1.2 Req 40:

**Test Type:** Capability

#### A.2.2.6 Tile Matrix

##### A.2.2.6.1 Data

##### A.2.2.6.1.1 Table Definition

**Test Case ID:** /opt/tiles/gpkg\_tile\_matrix/data/table\_def

**Test Purpose:** Verify that the gpkg\_tile\_matrix table exists and has the correct definition.

**Test Method:**

- 1) SELECT sql FROM sqlite\_master WHERE type = 'table' AND tbl\_name = 'gpkg\_tile\_matrix'
- 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 Annex C Table 23. Column order, check constraint and trigger definitions, and other column definitions in the returned sql are irrelevant.
- 4) Fail otherwise.

**Reference:** Clause 2.2.7.1.1 Req 41:

**Test Type:** Basic

##### A.2.2.6.1.2 Table Data Values

**Test Case ID:** /opt/tiles/gpkg\_tile\_matrix/data\_values\_table\_name

**Test Purpose:** Verify that values of the gpkg\_tile\_matrix table\_name column reference values in the gpkg\_contents table\_name column for rows with a data type of “tiles”.

**Test Method:**

- 1) SELECT table\_name FROM gpkg\_tile\_matrix
- 2) Not testable if returns an empty result set
- 3) SELECT table\_name FROM gpkg\_tile\_matrix tmm WHERE table\_name NOT IN (SELECT table\_name FROM gpkg\_contents gc WHERE tmm.table\_name = gc.table\_name AND gc.data\_type != 'tiles')
- 4) Fail if result set contains any rows
- 5) Pass otherwise

**Reference:** Clause 2.2.7.1.2 Req 42:

**Test Type:** Capability

**Test Case ID:** /opt/tiles/gpkg\_tile\_matrix/data/data\_values\_zoom\_level\_rows

**Test Purpose:** Verify that the gpkg\_tile\_matrix table contains a row record for each zoom level that contains one or more tiles in each tile pyramid 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 udt\_zoom FROM gpkg\_tile\_matrix 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>'
  - b. Fail if any gtmm\_zoom column value in the result set is NULL
- 4) Pass if no fails

**Reference:** Clause 2.2.7.1.2 Req 43:

**Test Type:** Capability

**Test Case ID:** /opt/tiles/gpkg\_tile\_matrix/data/data\_values\_zoom\_level

**Test Purpose:** Verify that zoom level column values in the gpkg\_tile\_matrix table are not negative.

**Test Method:**

- 1) SELECT zoom\_level FROM gpkg\_tile\_matrix
- 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 2.2.7.1.2 Req 44:

**Test Type:** Capability

**Test Case ID:** /opt/tiles/gpkg\_tile\_matrix/data/data\_values\_matrix\_width

**Test Purpose:** Verify that the matrix\_width values in the gpkg\_tile\_matrix table are valid.

**Test Method:**

- 1) SELECT matrix\_width FROM gpkg\_tile\_matrix
- 2) Not testable if returns an empty result set
- 3) SELECT min(matrix\_width) FROM gpkg\_tile\_matrix.
- 4) Fail if less than 1.
- 5) Pass otherwise.

**Reference:** Clause 2.2.7.1.2 Req 45:

**Test Type:** Capability

**Test Case ID:** /opt/tiles/gpkg\_tile\_matrix/data/data\_values\_matrix\_height

**Test Purpose:** Verify that the matrix\_height values in the gpkg\_tile\_matrix table are valid.

**Test Method:**

- 1) SELECT matrix\_height FROM gpkg\_tile\_matrix
- 2) Not testable if returns an empty result set
- 3) SELECT min(matrix\_height) FROM gpkg\_tile\_matrix.
- 4) Fail if less than 1.
- 5) Pass otherwise.

**Reference:** Clause 2.2.7.1.2 Req 46:

**Test Type:** Capability

**Test Case ID:** /opt/tiles/gpkg\_tile\_matrix/data/data\_values\_tile\_width

**Test Purpose:** Verify that the tile\_width values in the gpkg\_tile\_matrix table are valid.

**Test Method:**

- 1) SELECT tile\_width FROM gpkg\_tile\_matrix
- 2) Not testable if returns an empty result set
- 3) SELECT min(tile\_width) FROM gpkg\_tile\_matrix.
- 4) Fail if less than 1.
- 5) Pass otherwise.

**Reference:** Clause 2.2.7.1.2 Req 47:

**Test Type:** Capability

**Test Case ID:** /opt/tiles/gpkg\_tile\_matrix/data/data\_values\_tile\_height

**Test Purpose:** Verify that the tile\_height values in the gpkg\_tile\_matrix table are valid.

**Test Method:**

- 1) SELECT tile\_height FROM gpkg\_tile\_matrix
- 2) Not testable if returns an empty result set
- 3) SELECT min(tile\_height) FROM gpkg\_tile\_matrix.
- 4) Fail if less than 1.
- 5) Pass otherwise.

**Reference:** Clause 2.2.7.1.2 Req 48:

**Test Type:** Capability

**Test Case ID:** /opt/tiles/gpkg\_tile\_matrix/data/data\_values\_pixel\_x\_size

**Test Purpose:** Verify that the pixel\_x\_size values in the gpkg\_tile\_matrix table are valid.

**Test Method:**

- 1) SELECT pixel\_x\_size FROM gpkg\_tile\_matrix
- 2) Not testable if returns an empty result set
- 3) SELECT min(pixel\_x\_size) FROM gpkg\_tile\_matrix.
- 4) Fail if less than 0.
- 5) Pass otherwise.

**Reference:** Clause 2.2.7.1.2 Req 49:

**Test Type:** Capability

**Test Case ID:** /opt/tiles/gpkg\_tile\_matrix/data/data\_values\_pixel\_y\_size

**Test Purpose:** Verify that the pixel\_y\_size values in the gpkg\_tile\_matrix table are valid.

**Test Method:**

- 1) SELECT pixel\_y\_size FROM gpkg\_tile\_matrix
- 2) Not testable if returns an empty result set
- 3) SELECT min(pixel\_y\_size) FROM gpkg\_tile\_matrix.
- 4) Fail if less than 0.
- 5) Pass otherwise.

**Reference:** Clause 2.2.7.1.2 Req 50:

**Test Type:** Capability

**Test Case ID:** /opt/tiles/gpkg\_tile\_matrix/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 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
  - a. SELECT zoom\_level, pixel\_x\_size, pixel\_y\_size from gpkg\_tile\_matrix 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 2.2.7.1.2 Req 51:

**Test Type:** Capability

#### A.2.2.7 Tile Pyramid User Data

##### A.2.2.7.1 Data

##### A.2.2.7.1.1 Table Definition

**Test Case ID:** /opt/tiles/tile\_pyramid/data/table\_def

**Test Purpose:** Verify that multiple tile pyramids 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 2.2.8.1.1 Req 52:

**Test Type:** Basic

##### A.2.2.7.1.2 Table Data Values

**Test Case ID:** /opt/tiles/tile\_pyramid/data/data\_values\_zoom\_levels

**Test Purpose:** Verify that the zoom level column values in each tile pyramid user data table are within the range of zoom levels defined by rows in the gpkg\_tile\_matrix table.

**Test Method:**

- 1) SELECT DISTINCT table\_name AS <user\_data\_tiles\_table> FROM gpkg\_tile\_matrix
- 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 gpkg\_tile\_matrix 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 2.2.8.1.2 Req 53:

**Test Type:** Capability

**Test Case ID:** /opt/tiles/tile\_pyramid/data/data\_values\_tile\_column

**Test Purpose:** Verify that the tile\_column column values for each zoom level value in each tile pyramid user data table are within the range of columns defined by rows in the gpkg\_tile\_matrix table.

**Test Method:**

- 1) SELECT DISTINCT table\_name AS <user\_data\_tiles\_table> FROM gpkg\_tile\_matrix
- 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 gpkg\_tile\_matrix 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 2.2.8.1.2 Req 54:

**Test Type:** Capability

**Test Case ID:** /opt/tiles/tile\_pyramid\_data/data\_values\_tile\_row

**Test Purpose:** Verify that the tile\_row column values for each zoom level value in each tile pyramid user data table are within the range of rows defined by rows in the gpkg\_tile\_matrix table.

**Test Method:**

- 1) SELECT DISTINCT table\_name AS <user\_data\_tiles\_table> FROM gpkg\_tile\_matrix
- 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\_height AS gtmm\_height, udt.zoom\_level AS udt\_zoom, udt.tile\_row AS udt\_row FROM gpkg\_tile\_matrix 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 2.2.8.1.2 Req 55:

**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.8  
Table 31. Column order, check constraint and trigger definitions, and other column  
definitions in the returned sql are irrelevant.
- 4) Fail otherwise.

**Reference:** Clause 2.3.2.1.1 Req 56:

**Test Type:** Basic

##### A.2.3.1.1.2 Data Values

**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 2.3.2.1.2 Req 58:

**Test Type:** Capability

**Test Case ID:** /opt/schema/data\_columns/data/data\_values\_constraint\_name

**Test Purpose:** Verify that for each gpkg\_data\_columns row, the constraint\_name value is either NULL or a constraint\_name column value from the gpkg\_data\_column\_constraints table.

**Test Method:**

- 1) SELECT constraint\_name AS cn FROM gpkg\_data\_columns
- 2) Not testable if returns an empty result set
- 3) For each NOT NULL cn value from step 1
  - a. SELECT constraint\_name FROM gpkg\_data\_column\_constraints WHERE constraint\_name = cn
  - b. Fail if returns an empty result set
- 4) Pass if no fails

**Reference:** Clause 2.3.2.1.2 Req 59:

**Test Type:** Capability

**Test Case ID:** /opt/schema/data\_columns/data/data\_values\_constraint\_type

**Test Purpose:** Verify that for each gpkg\_data\_columns row, if the constraint\_name value is NOT NULL then the constraint\_type column value contains a constraint\_type column value from the gpkg\_data\_column\_constraints table for a row with a matching constraint\_name value.

**Test Method:**

- 1) SELECT constraint\_name AS cn, constraint\_type AS ct FROM gpkg\_data\_columns
- 2) Not testable if returns an empty result set
- 3) For each NOT NULL cn value from step 1
  - a. Fail if ct is NULL
  - b. If ct NOT NULL, SELECT constraint\_type FROM gpkg\_data\_column\_constraints WHERE constraint\_name = cn AND constraint\_type = ct
  - c. Fail if returns an empty result set
- 4) Pass if no fails

**Reference:** Clause 2.3.2.1.2 Req 59:

**Test Type:** Capability

### A.2.3.2 Data Column Constraints

#### A.2.3.2.1 Data

##### A.2.3.2.1.1 Table Definition

**Test Case ID:** /opt/schema/data\_column\_constraints/data\_table\_def

**Test Purpose:** Verify that the gpkg\_data\_column\_constraints table exists and has the correct definition.

**Test Method:**

- 1) SELECT sql FROM sqlite\_master WHERE type = 'table' AND tbl\_name = 'gpkg\_data\_column\_constraints'

- 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 Annex C Table 32. Column order, check constraint and trigger definitions, and other column definitions in the returned sql are irrelevant.
- 4) Fail otherwise.

**Reference:** Clause 2.3.3.1.1 Req 60:

**Test Type:** Basic

#### A.2.3.2.1.2 Data Values

**Test Case ID:** /opt/schema/data\_column\_constraints/data/data\_values\_constraint\_type

**Test Purpose:** Verify that the gpkg\_data\_column\_constraints constraint\_type column values are one of "range", "enum", or "glob".

**Test Method:**

- 1) SELECT constraint\_type AS ct FROM gpkg\_data\_column\_constraints
- 2) Not testable if returns an empty result set
- 3) For each ct value returned by step 1
  - a. Fail if ct NOT IN ("range", "enum", "glob").
- 4) Pass if no fails.

**Reference:** Clause 2.3.3.1.2 Req 61:

**Test Type:** Capability

**Test Case ID:**

/opt/schema/data\_column\_constraints/data/data\_values\_constraint\_names\_unique

**Test Purpose:** Verify that the gpkg\_data\_column\_constraints constraint\_name column values for constraint\_type values of "range", or "glob" are unique.

**Test Method:**

- 1) For each SELECT DISTINCT constraint\_name AS cn FROM gpkg\_data\_column\_constraints WHERE constraint\_type IN ('range', 'glob')
  - a. SELECT count(\*) FROM gpkg\_data\_column\_constraints WHERE constraint\_name = cn
  - b. Fail if count > 1
- 2) Pass if no fails.

**Reference:** Clause 2.3.3.1.2 Req 62:

**Test Type:** Capability

**Test Case ID:** /opt/schema/data\_column\_constraints/data/data\_values\_value\_for\_range

**Test Purpose:** Verify that the gpkg\_data\_column\_constraints value column values are NULL for rows with a constraint\_type value of "range".

**Test Method:**

- 1) SELECT constraint\_type AS ct, value AS v FROM gpkg\_data column\_constraints WHERE constraint\_type = 'range'
- 2) Not testable if returns an empty result set
- 3) For each v value returned by step 1
  - a. Fail if v IS NOT NULL
- 4) Pass if no fails.

**Reference:** Clause 2.3.3.1.2 Req 63:

**Test Type:** Capability

**Test Case ID:** /opt/schema/data\_column\_constraints/data/data\_values\_min\_max\_for\_range

**Test Purpose:** Verify that the gpkg\_data\_column\_constraints min column values are NOT NULL and less than the max column values for rows with a constraint\_type value of "range".

**Test Method:**

- 1) SELECT min, max FROM gpkg\_data column\_constraints WHERE constraint\_type = 'range'
- 2) Not testable if returns an empty result set
- 3) For each set of min and max values returned by step 1
  - a. Fail if min IS NULL
  - b. Fail if max IS NULL
  - c. Fail if min >= max
- 4) Pass if no fails.

**Reference:** Clause 2.3.3.1.2 Req 64:

**Test Type:** Capability

**Test Case ID:** /opt/schema/data\_column\_constraints/data/data\_values\_inclusive\_for\_range

**Test Purpose:** Verify that the gpkg\_data\_column\_constraints minIsInclusive and maxIsInclusive column values are NOT NULL and either 0 or 1 for rows with a constraint\_type value of "range".

**Test Method:**

- 1) SELECT minIsInclusive, maxIsInclusive FROM gpkg\_data column\_constraints WHERE constraint\_type = 'range'
- 2) Not testable if returns an empty result set
- 3) For each set of values returned by step 1
  - a. Fail if minIsInclusive IS NULL
  - b. Fail if maxIsInclusive IS NULL
  - c. Fail if minIsInclusive is NOT IN (0,1)
  - d. Fail if maxIsInclusive is NOT IN (0,1)
- 4) Pass if no fails.

**Reference:** Clause 2.3.3.1.2 Req 65:

**Test Type:** Capability

**Test Case ID:**

/opt/schema/data\_column\_constraints/data/data\_values\_min\_max\_inclusive\_for\_enum\_glob

**Test Purpose:** Verify that the gpkg\_data\_column\_constraints min, max, minIsInclusive and maxIsInclusive column values are NULL for rows with a constraint\_type value of "enum" or "glob".

**Test Method:**

- 1) SELECT min, max, minIsInclusive, maxIsInclusive FROM gpkg\_data column\_constraints WHERE constraint\_type IN ('enum','glob')
- 2) Not testable if returns an empty result set
- 3) For each set of values returned by step 1
  - a. Fail if min IS NOT NULL
  - b. Fail if max IS NOT NULL
  - c. Fail if minIsInclusive IS NOT NULL
  - d. Fail if maxIsInclusive IS NOT NULL
- 4) Pass if no fails.

**Reference:** Clause 2.3.3.1.2 Req 66:

**Test Type:** Capability

**Test Case ID:** /opt/schema/data\_column\_constraints/data/data\_values\_value\_for\_enum\_glob

**Test Purpose:** Verify that the gpkg\_data\_column\_constraints value column values are NOT NULL for rows with a constraint\_type value of "enum" or "glob".

**Test Method:**

- 1) SELECT value FROM gpkg\_data column\_constraints WHERE constraint\_type IN ('enum','glob')
- 2) Not testable if returns an empty result set
- 3) For each value returned by step 1
  - a. Fail if value IS NULL
- 4) Pass if no fails.

**Reference:** Clause 2.3.3.1.2 Req 67:

**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 68:

**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 15 in clause 2.4.2.1.2.

**Test Method:**

- 1) SELECT md\_scope FROM gpkg\_metadata
- 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 15 in clause 2.4.2.1.2
- 4) Pass if no fails

**Reference:** Clause 2.4.2.1.2 Req 69:

**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 70:

**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 71:

**Test Type:** Capability

**Test Case ID:** /opt/metadata/metadata\_reference/data/data\_values\_table\_name

**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 72:

**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 73:

**Test Type:** Capability

**Test Case ID:** /opt/metadata/metadata\_reference/data/data\_values\_row\_id\_value

**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 74:

**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 75:

**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 references a gpkg\_metadata id column value.

**Test Method:**

- 1) PRAGMA foreign\_key\_check('geometry\_columns')
- 2) Fail if returns any rows with a fourth column foreign key index value of 0

**Reference:** Clause 2.4.3.1.2 Req 75:

**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 77:

**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 36. Column order, check constraint and trigger definitions, and other column definitions in the returned sql are irrelevant.

4) Fail otherwise.

**Reference:** Clause 2.5.2.1.1 Req 78:

**Test Type:** Basic

#### A.2.5.1.1.2 Table Data Values

**Test Case ID:** /opt/extension\_mechanism/extensions/data/data\_values\_for\_extensions

**Test Purpose:** Verify that every extension of a GeoPackage is registered in a row in the gpkg\_extensions table

**Test Method:**

- 1) For each SELECT DISTINCT geometry\_type\_name FROM geometry\_columns
  - a. Fail if geometry\_type\_name IN Annex E Table 43 and gpkg\_extensions does not contain a row where extension\_name = gpkg\_geom\_<geometry\_type\_name>
  - b. Fail if geometry\_type\_name NOT IN Annex E Table 42 or Table 43 and gpkg\_extensions does not contain a row where the extension\_name does not begin with “gpkg” and the extension\_name ends with “\_geom\_<geometry\_type\_name>”
- 2) For each SELECT tbl\_name FROM sqlite\_master WHERE tbl\_name LIKE ‘rtree\_%’
  - a. Fail if gpkg\_extensions does not contain a row where extension\_name = “gpkg\_rtree\_index”
- 3) For each SELECT tbl\_name FROM sqlite\_master WHERE name LIKE ‘fgti\_%’
  - a. Fail if gpkg\_extensions does not contain a row where extension\_name = “gpkg\_geometry\_type\_trigger”
- 4) For each SELECT tbl\_name FROM sqlite\_master WHERE name LIKE ‘fgsi\_%’
  - a. Fail if gpkg\_extensions does not contain a row where extension\_name = “gpkg\_srs\_id\_trigger”
- 5) Do test /reg\_ext/tiles/zoom\_levels/data/zoom\_other\_ext\_row
- 6) Do test /reg\_ext/tiles/tile\_encoding\_webp/data/webp\_ext\_row
- 7) Do test /reg\_ext/tiles/tile\_encoding\_webp/data/tiff\_ext\_row
- 8) Do test /reg\_ext/tiles/tile\_encoding\_webp/data/nitf\_ext\_row
- 9) Pass if no fails

**Reference:** Clause 2.5.2.1.2 Req 79:

**Test Type:** Capability

**Test Case ID:** /opt/extension\_mechanism/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 2.5.2.1.2 Req 80:

**Test Type:** Capability

**Test Case ID:** /opt/extension\_mechanism/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 2.5.2.1.2 Req 81:

**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
- 3) For each row returned from step 1
  - a. Log pass if extension\_name is one of those listed in Annex J or Annex L through  
Annex P.
  - 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 2.5.2.1.2 Req 82:

**Test Type:** Capability

**Test Case ID:** /opt/extension\_mechanism/extensions/data/data\_values\_definition

**Test Purpose:** Verify that the definition column value contains or references extension documentation

**Test Method:**

- 1) SELECT definition FROM gpkg\_extensions
- 2) Not testable if returns an empty result set
- 3) For each row returned from step 1
  - a. Inspect if definition value is not like “Annex %”, or “http%” or [mailto:%](#) or “Extension Title%”
  - b. Fail if definition value does not contain or reference extension documentation
- 4) Pass if no fails

**Reference:** Clause 2.5.2.1.2 Req 83:

**Test Type:** Capability

**Test Case ID:** /opt/extension\_mechanism/extensions/data/data\_values\_scope

**Test Purpose:** Verify that the scope column value is “read-write” or “write-only”

**Test Method:**

- 1) SELECT scope FROM gpkg\_extensions
- 2) Not testable if returns an empty result set
- 3) For each row returned from step 1
  - a. Fail if value is not “read-write” or “write-only”
- 4) Pass if no fails

**Reference:** Clause 2.5.2.1.2 Req 84:

**Test Type:** Capability

### **A.3 Registered Extensions**

#### **A.3.1 Features**

##### **A.3.1.1 Geometry Types**

###### **A.3.1.1.1 Data**

###### **A.3.1.1.1.1 GeoPackage Extension Types**

**Test Case ID:**

/reg\_ext/features/geometry\_encoding/data/geopackage\_extension\_types/existing\_sparse\_data

**Test Purpose:** Verify that existing extended non-linear geometry types are stored in valid StandardGeoPackageBinary 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, log fail if GeoPackageBinary "X" type flag is 1
  - c. For each row from step a, if bytes 2-5 of cn.wkb as uint32 in endianness of gc.wkb byte 1 of cn from #1 are a geometry type value from Annex E Table 43, 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 cn.envelope.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 3.1.1.1.1 Req 85:

**Test Type:** Capability

#### Test Case ID:

/reg\_ext/features/geometry\_encoding/data/geopackage\_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 E, 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 GP headers containing big and little endian WKBs for 0-1 envelope contents indicator codes for every geometry type value from Annex E Table 43 without and with z and/or m values.
- 4) Fail otherwise

**Reference:** Clause 3.1.1.1.1 Req 85:

**Test Type:** Capability

#### A.3.1.1.1.2 GeoPackage Geometry Types -- Extensions Name

**Test Case ID:**

/reg\_ext/features/geometry\_encoding/data/geopackage\_extension\_types/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 E Table 43 used in a GeoPackage.

**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 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 two bytes of each geometry column value are “GP”, 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 WHERE 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 3.1.1.1.2 Req 86:

**Test Type:** Basic

#### A.3.1.1.1.3 GeoPackage Geometry Types -- Extensions Row

**Test Case ID:**

/reg\_ext/features/geometry\_encoding/data/geopackage\_extension\_types/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 3.1.1.1.3 Req 87:

**Test Type:** Capability

**A.3.1.2 User-Defined Geometry Types****A.3.1.2.1 Data****A.3.1.2.1.1 Extensions Encoding**

**Test Case ID:** /reg\_ext/features/geometry\_encoding/data/user\_defined-geometry\_types/existing\_sparse\_data

**Test Purpose:** Verify that existing extended geometry types not listed in Annex E are stored in valid ExtendedGeoPackageBinary 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 geometry\_type\_name NOT IN (all geometry types listed in Annex E) AND 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,
    - i. log fail if GeoPackageBinary "X" type flag is 0
    - ii. Otherwise log pass
- 6) Log pass if log contains pass and no fails

**Reference:** Clause 3.1.1.1.1 Req 88:

**Test Type:** Capability

**A.3.1.2.1.2 Extensions Name**

**Test Case ID:**

/reg\_ext/features/geometry\_encoding/data/user\_defined\_geometry\_types/extension\_name

**Test Purpose:** Verify that an extension name in the form <author>\_geom\_<gname> is defined for each extended geometry type not listed in Annex E used in a GeoPackage.

**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 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 two bytes of each geometry column value are "GP", 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 WHERE  
table\_name = result\_set\_table\_name AND column\_name =  
result\_set\_column\_name AND extension\_name NOT LIKE  
'gpkg\_%' and extension\_name LIKE '%\_geom\_' || <gtype>  
a. Fail if result set is empty  
b. Log pass otherwise

4) Pass if logged pass and no fails

**Reference:** Clause 3.1.1.1.2 Req 89:

**Test Type:** Basic

#### A.3.1.2.1.3 Extensions Row

**Test Case ID:**

/reg\_ext/features/geometry\_encoding/data/user\_defined\_geometry\_types/extension\_row

**Test Purpose:** Verify that the gpkg\_extensions table contains a row with an extension\_name in the form <author>\_geom\_<gname> for each table\_name and column\_name in the gpkg\_geometry\_columns table with a <gname> geometry\_type\_name.

**Test Method:**

Do test /reg\_ext/features/geometry\_encoding/data/extension\_encoding/extension\_name

**Reference:** Clause 3.1.1.1.3 Req 90:

**Test Type:** Capability

#### A.3.1.2.1.4 Geometry Columns Row

**Test Case ID:**

/reg\_ext/features/geometry\_encoding/data/user\_defined\_geometry\_types/geometry\_columns\_row

**Test Purpose:** Verify that the gpkg\_geometry\_columns table contains a row with a geometry\_type\_name in the form <author>\_geom\_<gname> for each feature table that contains user-defined geometry types specified in the gpkg\_extensions table.

**Test Method:**

- 1) SELECT extension\_name FROM gpkg\_extensions WHERE extension\_name LIKE  
'%\_geom\_%' AND extension\_name NOT LIKE 'gpkg\_geom\_%'
- 2) FOR EACH extension\_name from #1
  - a. SELECT \* FROM gpkg\_geometry\_columns WHERE geometry\_type\_name =  
extension\_name
  - b. Fail if returns an empty result set
- 3) Pass if no fails.

**Reference:** Clause 3.1.2.1.4 Req 91:

**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
- 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 Annex L 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 Annex L 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 Annex L 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 92:

**Test Type:** Capability

**Test Case ID:** /reg\_ext/features/spatial\_indexes/implementation/sql\_functions

**Test Purpose:** Verify the correct implementation of sql functions used in spatial indexes on feature table geometry columns.

**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 geometry type in Annex E, for an assortment of srs\_ids, for an assortment of coordinate values including empty geometries, without and with z and / or m values, in both big and little endian encodings:
  - a. SELECT 'Fail' FROM <gtype\_test> WHERE ST\_IsEmpty(geom.) != empty
  - b. SELECT 'Fail' FROM <gtype\_test> WHERE ST\_MinX(geom) != minx
  - c. SELECT 'Fail' FROM <gtype\_test> WHERE ST\_MaxX(geom) != maxx

- d. SELECT 'Fail' FROM <gtype\_test> WHERE ST\_MinY(geom) != miny
- e. SELECT 'Fail' FROM <gtype\_test> WHERE ST\_MaxY(geom) != maxy
- f.

3) Pass if no 'Fail' selected from step 2

**Reference:** Clause 3.1.3.1.1 Req 92:

**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 93:

**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:**

Do test /reg\_ext/features/spatial\_indexes/extension\_name

**Reference:** Clause 3.1.3.1.3 Req 94:

**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 = 'fgtu\_' || 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 Annex M 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 Annex M 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 95:**Test Type:** Capability**Test Case ID:** /reg\_ext/features/geometry\_type\_triggers/implementation /sql\_functions**Test Purpose:** Verify the correct implementation of sql functions used in geometry type triggers on feature table geometry columns.**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 E, 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 GPKG\_IsAssignable(gtype, atype) = 0
  - b. SELECT 'Fail' FROM <gtype\_test> WHERE GPKG\_IsAssignable(ntype, atype) = 1
  - c. SELECT 'Fail' FROM <gtype\_test> WHERE ST\_GeometryType(geom) != atype
- 3) Pass if no 'Fail' selected from step 2

**Reference:** Clause 3.1.4.1.1 Req 95:

**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:**

- 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 96:

**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:**

Do test /reg\_ext/features/geometry\_type\_triggers/extension\_name

**Reference:** Clause 3.1.4.1.3 Req 97:

**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
  - c. Fail if sql != result of populating the first trigger template in Annex N 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 Annex N 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 98:

**Test Type:** Capability

**Test Case ID:** /reg\_ext/features/srs\_id\_triggers/implementation/sql\_functions

**Test Purpose:** Verify the correct implementation of sql functions used in srs\_id triggers on feature table geometry columns.

**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 geometry type in Annex E, 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
- 3) Pass if no 'Fail' selected from step 2

**Reference:** Clause 3.1.5.1.1 Req 98:

**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 99:

**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:**

Do test /reg\_ext/features/srs\_id\_triggers/extension\_name

**Reference:** Clause 3.1.5.1.3 Req 100:

**Test Type:** Capability

### A.3.2 Tiles

#### 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 factors of two zoom intervals.

**Test Method:**

- 1) SELECT table\_name FROM gpkg\_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 gpkg\_tile\_matrix 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 factors 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 3.2.1.1.1 Req 101:

**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 factors of two zoom intervals are registered using the “gpkg\_zoom\_other” extension name.

**Test Method:**

Do test /reg\_ext/tiles/zoom\_levels/data/zoom\_other\_ext\_name

**Reference:** Clause 3.2.1.1.2 Req 102:

**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

**Test Purpose:** Verify that the “gpkg\_webp” extensions name is used to register WEBP tile encoding implementations.

**Test Method:**

- 1) SELECT table\_name FROM gpkg\_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 103:

**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:**

Do test /reg\_ext/tiles/tile\_encoding\_webp/data/webp\_ext\_name

**Reference:** Clause 3.2.2.2.2 Req 104:

**Test Type:** Capability

## **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 OGC® GeoPackage Encoding Standard 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 [9][10][11] and SQL-MM (Spatial) [12] 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 [16] and ISO metadata [28], 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 Encoding Standard to the Open Geospatial Consortium Inc. as a Request For Comment (RFC).

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

## f) U.S. National Geospatial Intelligence Agency

**B.3 Revision history**

| <b>Date</b> | <b>Rel</b> | <b>Editor</b> | <b>Paragraph modified</b> | <b>Description</b>                                                                                                                                                                                                                                                                                                             |
|-------------|------------|---------------|---------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 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 (). 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 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<br>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 superfluous 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                                                                                                               |
| 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                                                    |
| 2013-05-07 | 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<br>add Annex M<br>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 Daisey | 7.3.2                                                     | Add SQLite configuration requirements                                               |
| 2013-05-22 | R5 | Paul        | 6.3, 7.2                                                  | Require only gpkg_contents and                                                      |

|            |    |             |                                                                                                                                     |                                                                                                       |
|------------|----|-------------|-------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------|
|            |    | Daisey      |                                                                                                                                     | 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, Introduction, Clauses 1-5 | Remove all forward material except title page, submitting orgs, and introduction, and put in annexes. |

|            |    |                      |                                              |                                                                                                  |
|------------|----|----------------------|----------------------------------------------|--------------------------------------------------------------------------------------------------|
| 2013-06-07 | R6 | Paul Daisey          | Old Clauses 6,7<br>-> 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 Daisey          | All                                          | Change requirement statement format to <b>Req # s SHALL o in bold italic</b>                     |
| 2013-06-28 | R6 | Paul                 | Annex B                                      | Update definition of Empty GeoPackage,                                                           |

|            |    |             |                                    |                                                                                                                                                                                             |
|------------|----|-------------|------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|            |    | Daisey      |                                    | add definition of Valid GeoPackage                                                                                                                                                          |
| 2013-06-28 | R6 | Paul Daisey | Figure 1, 2.2.7, Annex C, F        | Change tile_matrixI_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                                                                                                                                   |
| 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<br>Table 3,<br>2.1.6.1.2,<br>Annex A, C   | Nullable gpkg_contents columns<br>One geometry column per feature table.                                                   |
| 2013-09-09 | R8 | Paul Daisey | Clause 3                                            | Add new Req 67 re gpkg_extensions table                                                                                    |
| 2013-09-09 | R8 | Paul Daisey | 2.1.6.1.1                                           | remove footnote 1                                                                                                          |
| 2013-09-09 | R8 | Paul Daisey | 2.1.6.1.1,<br>Annex C                               | change no to yes for null in Table 6, 27<br>except for id column                                                           |
| 2013-09-09 | R8 | Paul Daisey | 1.1.3.1.1                                           | Req 12 add gpkg_prefix to<br>gpkg_spatial_ref_sys                                                                          |
| 2013-09-09 | R8 | Paul Daisey | 2.2                                                 | remove "with compression" and "without<br>compression from footnote 1                                                      |
| 2013-09-09 | R8 | Paul Daisey | 2.6.1.1.1                                           | "an all of" changed to "all"                                                                                               |
| 2013-09-09 | R8 | Paul Daisey | 2.2.6.1.2,<br>Annex A                               | Add gpkg_prefix to<br>gpkg_tile_matrix_metadata                                                                            |
| 2013-09-09 | R8 | Paul Daisey | 2.2.7.1.1                                           | Space between 8and                                                                                                         |
| 2013-09-09 | R8 | Paul Daisey | 2.2.6.1.1                                           | Table 7 {RasterLayerName}_tiles change to<br>Tile Matrix User Data Table Name                                              |
| 2013-09-09 | R8 | Paul Daisey | 3.1.3.1.2,<br>3.2.1.1.2,<br>Annex C                 | Change gpkg_extension to gpkg_extensions                                                                                   |
| 2013-09-09 | R8 | Paul Daisey | 3.2.3.1.1                                           | Remove [29] from GeoTiff, change [31] to<br>[29] for NGA Implementation Profile                                            |
| 2013-09-09 | R8 | Paul Daisey | Annex A                                             | change "first three bytes of each gc are not<br>"GPB"" to "first two bytes of each gc are not<br>"GP"                      |
| 2013-09-09 | R8 | Paul Daisey | Annex A                                             | Change (1,2,3) to (0,1,2)                                                                                                  |
| 2013-09-09 | R8 | Paul Daisey | Annex D                                             | add INTEGER return type to ST_ISEmpty,<br>ST_Is3D, ST_ISMeasured functions                                                 |
| 2013-09-09 | R8 | Paul Daisey | 2.1.1                                               | Add "more"                                                                                                                 |
| 2013-09-09 | R8 | Paul Daisey | Annex C                                             | Add the following constraint to the<br>gpkg_geometry_columns table:<br>CONSTRAINT uk_gc_table_name<br>UNIQUE (table_name), |
| 2013-09-09 | R8 | Paul Daisey | 3.1.4.1.2,<br>3.1.5.1.2                             | Change geopackage_extension to<br>gpkg_extensions                                                                          |

|            |    |             |                                                     |                                                                                                                                                                                                                                                                                                                                                                          |
|------------|----|-------------|-----------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2013-09-18 | R8 | Paul Daisey | 2.6.1, Annex A, Annex C                             | Add column 'scope text not null' to gpkg_extensions                                                                                                                                                                                                                                                                                                                      |
| 2013-09-18 | R8 | Paul Daisey | 2.5                                                 | Reword clause 2.5 to preclude reference to other data tables by rows in gpkg_geometry_columns or gpkg_tile_matrix_metadata tables.<br>Add data_type constraint to Req 19.<br>Add data_type constraint to Annex A tests.                                                                                                                                                  |
| 2013-09-18 | R8 | Paul Daisey | 3.1, 3.3, Annex A, Annex D                          | Rename clause 3.3 Other Tables<br>Move clause 3.1.1 Geometry Encoding to clause 3.3.1 Other Geometry Encoding<br>Move clause 3.2.5 Tile Encoding Other to clause 3.3.2 Other Tile Encoding<br>Rename clause 3.3.1 Other Trigger to clause 3.3.3 Other Trigger<br>Renumber contained requirements<br>Add footnote to Annex D table<br>Move corresponding tests in Annex A |
| 2013-09-24 | R8 | Paul Daisey | Annex E                                             | Replace all occurrences of 'rowid' with '<i>' in trigger definitions.                                                                                                                                                                                                                                                                                                    |
| 2013-09-24 | R8 | Paul Daisey | All                                                 | Update normative [xx] and bibliographic [Byy] references                                                                                                                                                                                                                                                                                                                 |
| 2013-09-25 | R8 | Paul Daisey | 3.1, Annex A, Annex D, Annex F                      | Change geometry_columns to gpkg_geometry_columns and geometry_type to geometry_type_name                                                                                                                                                                                                                                                                                 |
| 2013-09-25 | R8 | Paul Daisey | 2.2.3, 3.2.1, new Annex I                           | Change “power” to “factor” in and add text to zoom level descriptions and requirements.                                                                                                                                                                                                                                                                                  |
| 2013-09-25 | R8 | Paul Daisey | 2.1.6, Annex A, Annex C                             | Add required data types for vector feature user data table column definitions.                                                                                                                                                                                                                                                                                           |
| 2013-09-26 | R8 | Paul Daisey | New 2.3.2, Annex A, Annex C                         | Add gpkg_data_column_constraints table                                                                                                                                                                                                                                                                                                                                   |
| 2013-09-26 | R8 | Paul Daisey | 1.1.3, 2.2.1, 2.2.6, 2.2.7, 2.2.8, Annex A, Annex C | Add gpkg_tile_matrix_set, rename gpkg_tile_matrix_metadata to gpkg_tile_matrix, standardize tile pyramid, tile matrix set and tile matrix term usage.                                                                                                                                                                                                                    |
| 2013-09-27 | R8 | Paul Daisey | Figure 1                                            | Added gpkg_tile_matrix_set, gpkg_data_columns_constraints to Figure 1                                                                                                                                                                                                                                                                                                    |
| 2013-09-27 | R8 | Paul Daisey | 2.1.5.1                                             | Replace normative text in 2.1.5.1.1 with new requirement 19 in 2.1.5.1.2.                                                                                                                                                                                                                                                                                                |
| 2013-09-27 | R8 | Paul Daisey | All                                                 | Clarify which tables may be implemented as views or extended with additional columns.                                                                                                                                                                                                                                                                                    |
| 2013-09-27 | R8 | Paul Daisey | 2.4.2.1.1                                           | Drop md_standard_uri default value, add footnote and text paragraph.                                                                                                                                                                                                                                                                                                     |
| 2013-09-27 | R8 | Paul Daisey | All                                                 | Specify case sensitivity for required column values.                                                                                                                                                                                                                                                                                                                     |

|            |    |             |                                     |                                                                                                                      |
|------------|----|-------------|-------------------------------------|----------------------------------------------------------------------------------------------------------------------|
| 2013-09-27 | R8 | Paul Daisey | All                                 | Change SQL trigger action from ROLLBACK to ABORT                                                                     |
| 2013-10-02 | R9 | Paul Daisey | 1.1.2.1.1, Annex A                  | Add Null column to table 2 per table 22, remove default value, foreign and unique key tests from Annex A A.1.1.2.1.1 |
| 2013-10-02 | R9 | Paul Daisey | All                                 | Misc. Editorial corrections.                                                                                         |
| 2013-10-02 | R9 | Paul Daisey | Title Page                          | Changed title from Implementation Specification to Encoding Standard per Carl Reed email.                            |
| 2013-10-04 | R9 | Paul Daisey | 1.1.1.1.3, Annex A                  | Add new Req 4 requirement for SQLite PRAGMA foreign_key_check()                                                      |
| 2013-10-04 | R9 | Paul Daisey | All                                 | Disallow addition of columns to all gpkg tables except user data feature tables.                                     |
| 2013-10-05 | R9 | Paul Daisey | Introduction, All, Annex B          | GeoPackage replaces GeoPackage file                                                                                  |
| 2013-10-05 | R9 | Paul Daisey | Introduction, 1.1.1.1.1, Annex B    | Add Extended GeoPackage definition                                                                                   |
| 2013-10-05 | R9 | Paul Daisey | Introduction Figure 1, 2.6, Annex C | Add definition column to gpkg_extensions table.                                                                      |
| 2013-10-05 | R9 | Paul Daisey | Introduction, Annex B               | Add GeoPackage SQLite Configuration definition                                                                       |
| 2013-10-05 | R9 | Paul Daisey | Introduction, Annex B               | Modify definition of GeoPackage SQLite Extension                                                                     |
| 2013-10-05 | R9 | Paul Daisey | 1                                   | Revise Container Definition, add new Req 1 and Req 3                                                                 |
| 2013-10-05 | R9 | Paul Daisey | 1.1.1.2.2                           | Remove rows from Table 1 SQLite Configuration                                                                        |
| 2013-10-05 | R9 | Paul Daisey | 2.1.3.2                             | Drop clause. Required SQL functions to be defined in extension template documents.                                   |
| 2013-10-05 | R9 | Paul Daisey | 2.5.1                               | New Introduction clause for Extension Mechanism                                                                      |
| 2013-10-05 | R9 | Paul Daisey | 2.5.2.1.2                           | New Req 79, Req 83, revised Req 82, added explanatory text for Req 84, deleted Table 47                              |
| 2013-10-07 | R9 | Paul Daisey | 1.1.1.2.2                           | Move foreign key rows from Table 9 Safe GeoPackage to Table 1 SQLite Configuration                                   |
| 2013-10-07 | R9 | Paul Daisey | 2.5.2.1                             | Extend Req 79 with semantic equivalence of empty and missing gpkg_extensions table.                                  |
| 2013-10-07 | R9 | Paul Daisey | 2.5.2.2                             | Drop Clause. Required SQL Configuration settings to be defined in extension template documents.                      |
| 2013-10-07 | R9 | Paul        | 3.1.2                               | New clause for Geometry Encoding                                                                                     |

|            |    |             |                                     |                                                                                                                                                                 |
|------------|----|-------------|-------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
|            |    | Daisey      |                                     | Extension                                                                                                                                                       |
| 2013-10-07 | R9 | Paul Daisey | Annex J - S                         | New Extension Template and Extension Annexes                                                                                                                    |
| 2013-10-08 | R9 | Paul Daisey | 1.1.1.2.2, Annex M, N, O            | Prohibit all SQLITE_OMIT_* compile time options instead of specific ones.                                                                                       |
| 2013-10-09 | R9 | Paul Daisey | 3.*.1.1                             | Simplify extension prerequisite descriptions, reference extension annexes.                                                                                      |
| 2013-10-09 | R9 | Paul Daisey | Annexes J - S                       | Style subheadings as subtitles, add clause references                                                                                                           |
| 2013-10-09 | R9 | Paul Daisey | 2.1.3, Annex L                      | Replace GeoPackageBinary definition with StandardGeoPackageBinary definition.                                                                                   |
| 2013-10-09 | R9 | Paul Daisey | Introduction, 2.3.2, 2.3.3, Annex C | 1. Drop constraint_type column from gpkg_data_columns, Req 60<br>2. Add Req 61<br>3. Change description of constraint name.<br>4. Revise Figure 1 Table Diagram |
| 2013-10-09 | R9 | Paul Daisey | 2.5.2.1.2                           | Add guidance for extension template                                                                                                                             |
| 2013-10-09 | R9 | Paul Daisey | 2.1.6.1.2                           | Add footnote re further restrictions on the geometry types that are allowed in a feature table                                                                  |
| 2013-10-09 | R9 | Paul Daisey | 2.1.3.1.1                           | Add footnote that WKB geometries are not geographic / geodesic                                                                                                  |
| 2013-10-10 | R9 | Paul Daisey | 1.1.1.1.2                           | Change Req 3 SQLite database header required contents to include version number.                                                                                |
| 2013-10-10 | R9 | Paul Daisey | Annex A                             | Add and revise tests in accordance with other R9 draft changes.                                                                                                 |
| 2013-10-10 | R9 | Paul Daisey | 2.3.3, Annex A                      | Add unique key on gpkg_data_column_constraints constraint_name, constraint_type, constraint_value columns; Add new Req 63, ATS test                             |
| 2013-10-15 | R9 | Paul Daisey | Annex D                             | Drop D.1 Geometry Column Triggers                                                                                                                               |
| 2013-10-15 | R9 | Paul Daisey | Annex F                             | Drop Annex F                                                                                                                                                    |
| 2013-10-15 | R9 | Paul Daisey | 2.3.3.1.2                           | Remove first sentence on constrain name column values.                                                                                                          |
| 2013-10-15 | R9 | Paul Daisey | Introduction, 2.3.3.1.1, Annex C    | Add description column to gpkg_data_column_constraints table.                                                                                                   |
| 2013-10-15 | R9 | Paul Daisey | 2.3.3.1.2                           | Drop requirement 61                                                                                                                                             |
| 2013-10-15 | R9 | Paul Daisey | 2.1.6.1.2                           | Drop footnote 2 on p.13                                                                                                                                         |
| 2013-10-15 | R9 | Paul        | 2.1.6.1.1                           | Delete DECIMAL_TEXT row from Table 6                                                                                                                            |

|            |    |             |                          |                                                                                                                                                                      |
|------------|----|-------------|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|            |    | Daisey      |                          |                                                                                                                                                                      |
| 2013-10-15 | R9 | Paul Daisey | 1.1.1.1.1                | Move to 1.1.1.1.3, renumber requirements.                                                                                                                            |
| 2013-10-15 | R9 | Paul Daisey | 2.1.6.1.1, 1.1.1.1.3     | Move Req 30 and Table 6 to Req5 and Table 1                                                                                                                          |
| 2013-10-15 | R9 | Paul Daisey | 2.4.3.1.1, Annex C       | Change timestamp column from TEXT to DATETIME data type                                                                                                              |
| 2013-10-15 | R9 | Paul Daisey | All                      | Change data types to those specified in Table 1                                                                                                                      |
| 2013-10-15 | R9 | Paul Daisey | 2.3.3.1.1                | Add NUMERIC data type footnote for table <code>gpkg_data_column_constraints</code>                                                                                   |
| 2013-10-15 | R9 | Paul Daisey | All                      | Misc editorial changes                                                                                                                                               |
| 2013-10-22 | R9 | Paul Daisey | 3.2.3, 3.2.4, Annex A    | Remove TIFF and NITF tile encodings                                                                                                                                  |
| 2013-10-22 | R9 | Paul Daisey | New 3.1.2.1.4 Annex A    | New Req 91                                                                                                                                                           |
| 2013-10-22 | R9 | Paul Daisey | Annex K                  | Rewrite requirement description                                                                                                                                      |
| 2013-10-22 | R9 | Paul Daisey | 3.1.1, 3.1.2, Annex J, K | Renamed                                                                                                                                                              |
| 2013-10-22 | R9 | Paul Daisey | All                      | Misc editorial changes                                                                                                                                               |
| 2013-10-22 | R9 | Paul Daisey | Introduction, Annex B    | Replace detailed table diagram in Introduction with an overview diagram; move detailed table diagram to Annex B, add minimal features and tiles diagrams to Annex B. |
| 2013-10-24 | R9 | Paul Daisey | 1.1.1.1.3                | Specify 8601 formats for DATE and DATETIME in Table 1                                                                                                                |
| 2013-10-24 | R9 | Paul Daisey | 1.1.1.1.3                | {{(size)}} specs for BLOB and TEXT data types in Table 1                                                                                                             |
| 2013-10-24 | R9 | Paul Daisey | 2.1.3.1.1 Table 5        | Bit layout of flags byte and flag bits use -- additional text                                                                                                        |
| 2013-10-24 | R9 | Paul Daisey | All                      | Misc editorial changes                                                                                                                                               |
| 2013-10-24 | R9 | Paul Daisey | Annex Q, R               | Removed                                                                                                                                                              |
| 2013-10-24 | R9 | Paul Daisey | All, Annex S, T          | Remove unused normative [xx]and bibliographic [Byy] references, update remaining ones                                                                                |
| 2013-11-11 | R9 | Paul Daisey | All                      | Misc editorial changes, mostly formatting.                                                                                                                           |
| 2013-11-12 | R9 | Paul Daisey | Annex C, G               | Correct <code>gpkg_metadata</code> table definition SQL default values, metadata column type from BLOB to TEXT                                                       |

|            |    |                              |                                       |                                                                                                                                                                       |
|------------|----|------------------------------|---------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2013-11-12 | R9 | Paul Daisey                  | 1.1.1.1.1                             | Add footnote (3) to Req 2                                                                                                                                             |
| 2013-11-12 | R9 | Paul Daisey                  | Annex L, O                            | Add <a href="http://www.sqlite.org/rtree.html">http://www.sqlite.org/rtree.html</a>                                                                                   |
| 2013-11-12 | R9 | Paul Daisey                  | 2.2.6.1.1,<br>2.2.6.1.2               | Cut sentence from 2.2.6.1.1, paste revised version and additional sentence describing origin georeferenced coordinates to 2.2.6.1.2.                                  |
| 2013-11-12 | R9 | Paul Daisey                  | 2.3.3.1.2                             | Add a new third sentence in first paragraph; add new Table 13 example                                                                                                 |
| 2013-11-12 | R9 | Paul Daisey                  | 1.1.3.1.1,<br>2.1.3.1.1,<br>2.2.6.1.1 | Add x/y axis definitions to min/max x/y column descriptions and WKB geometry description.                                                                             |
| 2013-11-13 | R9 | Paul Daisey                  | 1.1.1.1.1                             | Revise footnote (3) to Req 2 to include byte values instead of long value.                                                                                            |
| 2013-11-13 | R9 | Jeff Yutzler,<br>Paul Daisey | All                                   | Misc editorial changes, mostly formatting.                                                                                                                            |
| 2013-11-18 | R9 | Jeff Yutzler,<br>Paul Daisey | All                                   | Misc editorial changes, mostly formatting.                                                                                                                            |
| 2013-11-18 | R9 | Paul Daisey                  | 1,<br>2.1.6.1.1,<br>2.2.8.1.1         | Add “and table constraints” in clause 1, make INTEGER PRIMARY KEY AUTOINCREMENT explicit in Req 29 and Req 52 to avoid future “WITHOUT ROWID” SQLite v3.8.2 problems. |
| 2013-11-19 | R9 | Jeff Yutzler,<br>Paul Daisey | All                                   | Misc formatting changes.                                                                                                                                              |

#### **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 OGC® Implementation Standards**

None at present.

#### **B.6 Potential Future Work**

- MAY investigate GeoPackage implementation on SQLite version 4 [B25].
- 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).
- 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 that contains a `gpkg_spatial_ref_sys` table, a `gpkg_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 2.2.8 exist but contain no rows.

### **B.8.2 Extended GeoPackage**

a GeoPackage that contains any additional data elements (tables or columns) or SQL constructs (data types, indexes, constraints or triggers) that are not specified in this encoding standard.

### **B.8.3 geolocate**

identify a real-world geographic location

### **B.8.4 GeoPackage**

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 Configuration**

consists of the SQLite 3 software library and a set of compile- and runtime configurations options.

### **B.8.6 GeoPackage SQLite Extension**

a SQLite loadable extension that MAY provide SQL 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.

### **B.8.7 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.8.8 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.8.9 Valid GeoPackage**

a GeoPackage 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:

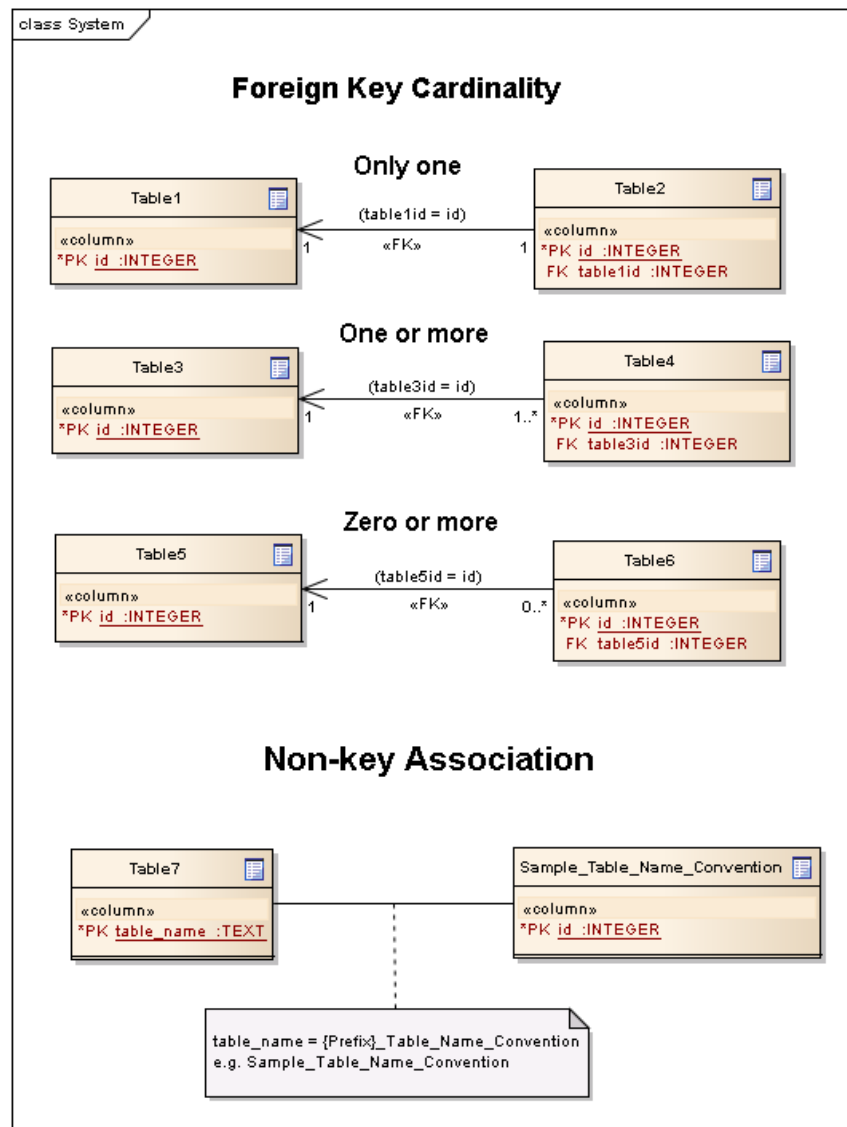
|       |                                                                    |
|-------|--------------------------------------------------------------------|
| ACID  | Atomic, Consistent, Isolated, and Durable                          |
| ASCII | American Standard Code for Information Interchange                 |
| API   | Application Program Interface                                      |
| BLOB  | Binary Large Object                                                |
| CLI   | Call-Level Interface                                               |
| COTS  | Commercial Off The Shelf                                           |
| DEM   | Digital Elevation Model                                            |
| 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                              |
| NIIRS | National Imagery Interpretability Rating Scale                     |
| 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 |
| 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) [B14] static structure diagrams. The UML notations used in this standard for RDBMS tables in a GeoPackage are described in Figure 3 below.

Figure 3 UML Notation for RDBMS Tables



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

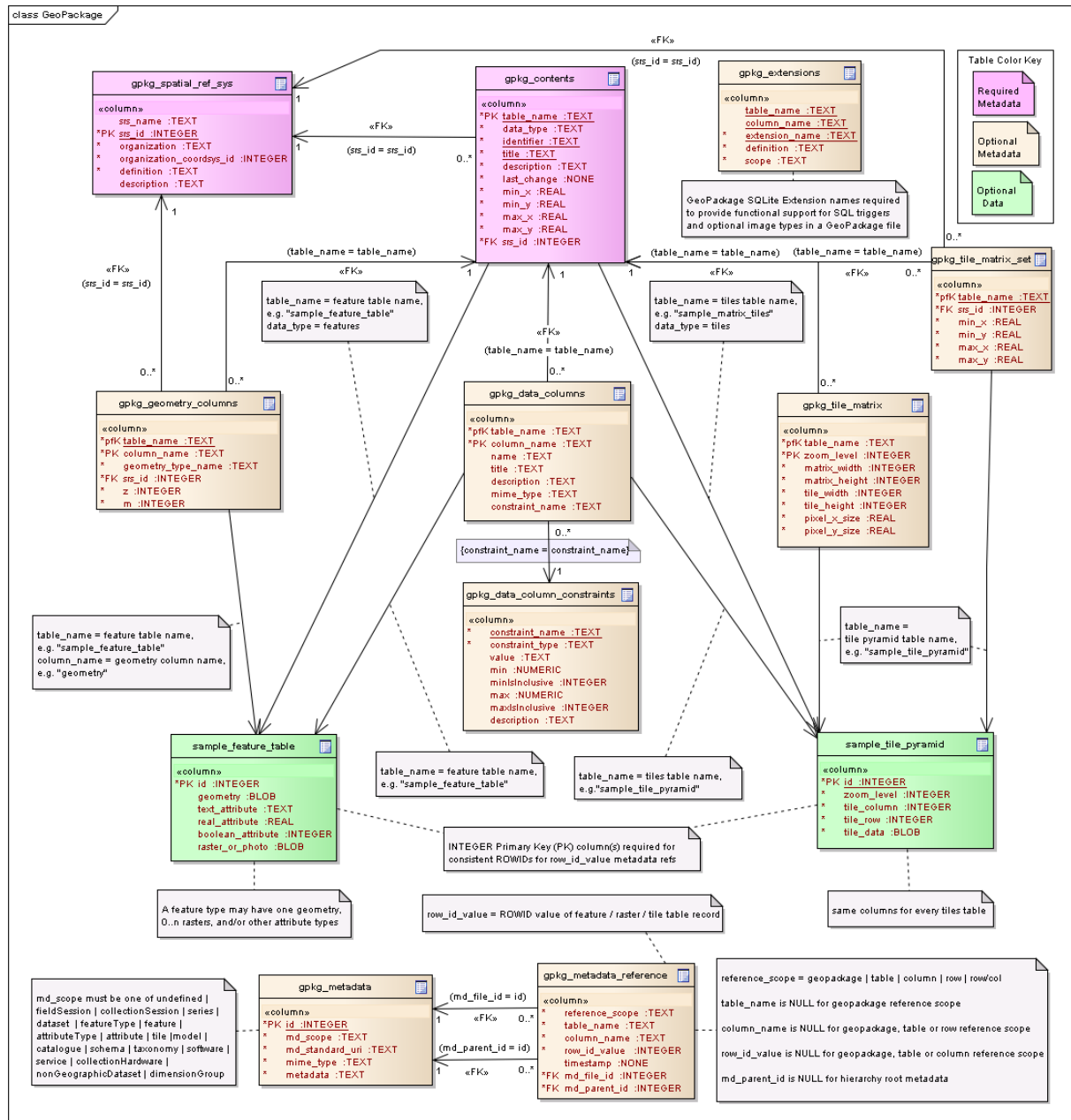
- <<table>> An instantiation of a UML class as an RDBMS table.
- <<column>> An instantiation of a UML attribute as an RDBMS table column.

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.
- 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.

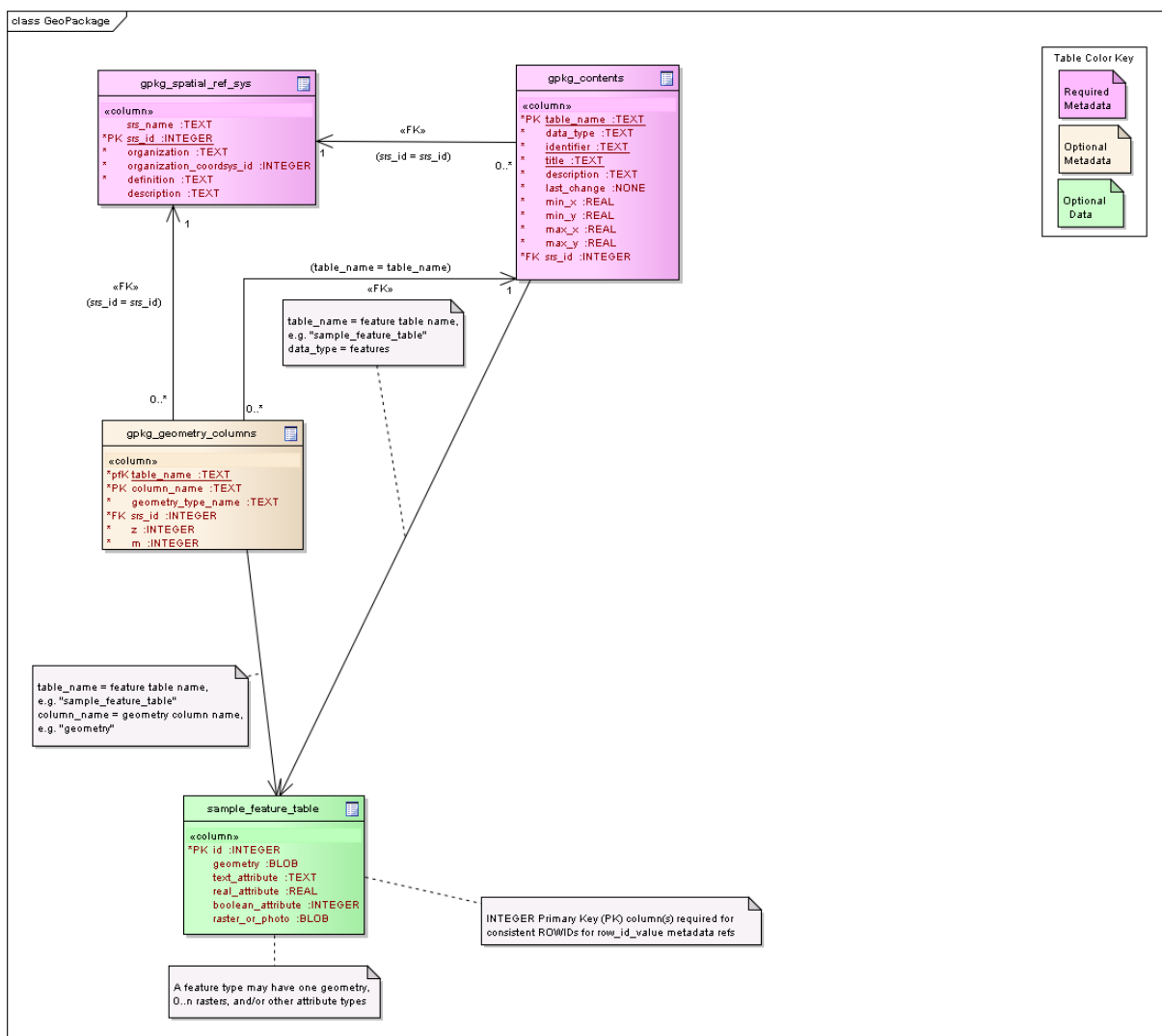
## B.11 GeoPackage Tables Detailed Diagram

Figure 4 GeoPackage Tables Details



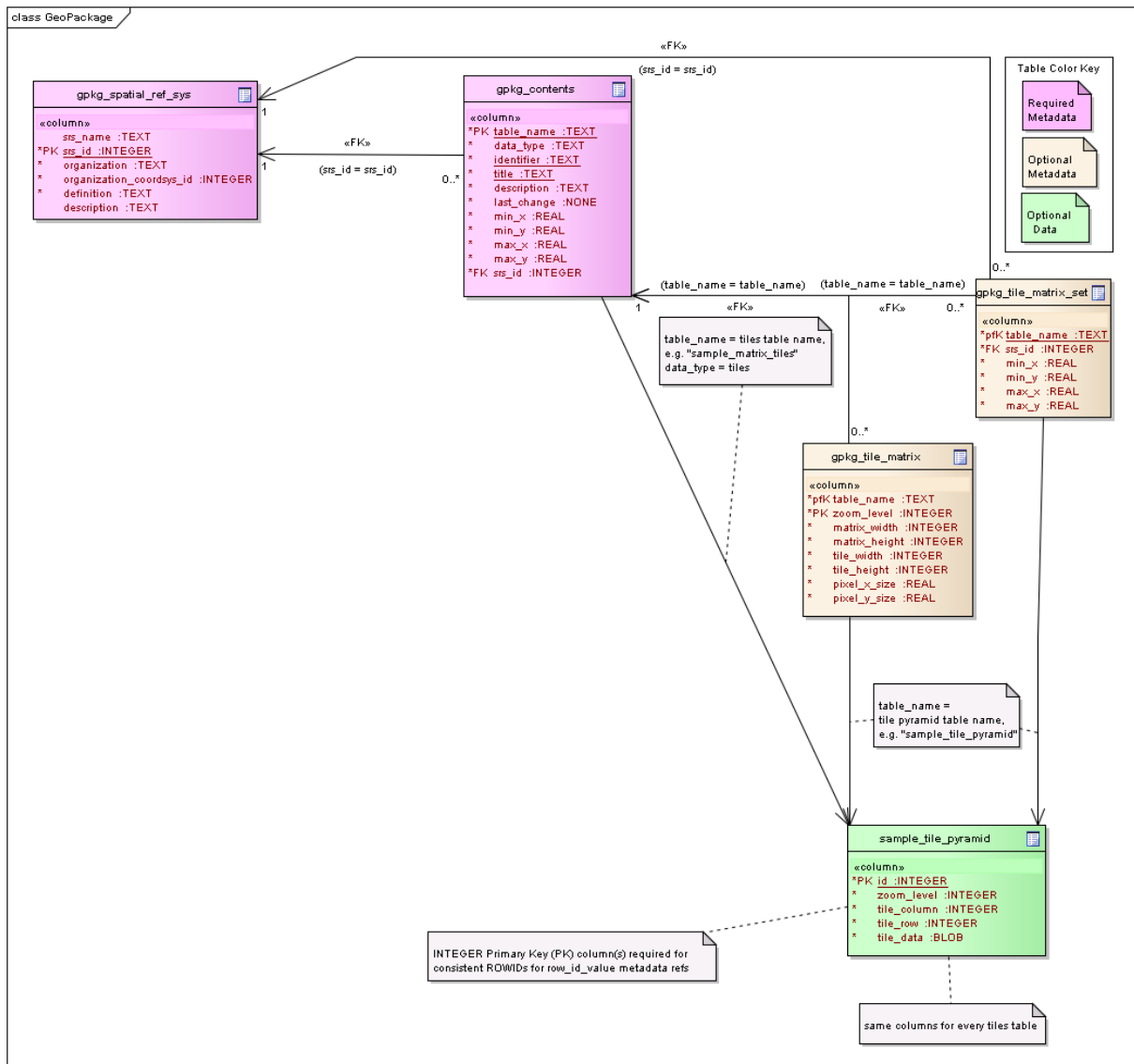
## B.12 GeoPackage Minimal Tables for Features Diagram

Figure 5 GeoPackage Minimal Tables for Features



## B.13 GeoPackage Minimal Tables for Tiles Diagram

Figure 6 GeoPackage Minimal Tables for Tiles



## Annex C Table Definition SQL (Normative)

### C.1 gpkg\_spatial\_ref\_sys

**Table 18 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_id INTEGER NOT NULL,
 definition TEXT NOT NULL,
 description TEXT
)
```

**Table 19 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 20 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 21 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 DATETIME NOT NULL DEFAULT
 (strftime('%Y-%m-%dT%H:%M:%fZ',CURRENT_TIMESTAMP)),
 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\_geometry\_columns

**Table 22 gpkg\_geometry\_columns Table Definition SQL**

```
CREATE TABLE gpkg_geometry_columns (
 table_name TEXT NOT NULL,
 column_name TEXT NOT NULL,
 geometry_type_name TEXT NOT NULL,
 srs_id INTEGER NOT NULL,
 z TINYINT NOT NULL,
 m TINYINT NOT NULL,
 CONSTRAINT pk_geom_cols PRIMARY KEY (table_name, column_name),
 CONSTRAINT uk_gc_table_name UNIQUE (table_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 23 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;
```

**Table 24 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,
 code4name1(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.4 sample\_feature\_table (Informative)

<sup>1</sup> Implementer must provide code4name(geometry\_type\_name) SQL function

**Table 25 sample\_feature\_table Table Definition SQL**

```
CREATE TABLE sample_feature_table (
 id INTEGER PRIMARY KEY AUTOINCREMENT,
 geometry GEOMETRY,
 text_attribute TEXT,
 real_attribute REAL,
 boolean_attribute BOOLEAN,
 raster_or_photo BLOB
)
```

**C.5 gpkg\_tile\_matrix\_set****Table 26 gpkg\_tile\_matrix\_set Table Creation SQL**

```
CREATE TABLE gpkg_tile_matrix_set (
 table_name TEXT NOT NULL PRIMARY KEY,
 srs_id INTEGER NOT NULL
 min_x DOUBLE NOT NULL,
 min_y DOUBLE NOT NULL,
 max_x DOUBLE NOT NULL,
 max_y DOUBLE NOT NULL,
 CONSTRAINT fk_gtms_table_name FOREIGN
 KEY (table_name) REFERENCES gpkg_contents(table_name),
 CONSTRAINT fk_gtms_srs FOREIGN KEY (srs_id) REFERENCES
 gpkg_spatial_ref_sys (srs_id)
)
```

**C.6 gpkg\_tile\_matrix****Table 27 gpkg\_tile\_matrix Table Creation SQL**

```
CREATE TABLE gpkg_tile_matrix (
 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),
 CONSTRAINT fk_tmm_table_name FOREIGN KEY (table_name)
 REFERENCES gpkg_contents(table_name))
```

**Table 28 EXAMPLE: gpkg\_tile\_matrix Insert Statement**

```
INSERT INTO gpkg_tile_matrix VALUES (
 "sample_tile_pyramid",
 0,
 1,
 1,
 512,
```

```
512,
2.0,
2.0);
```

### C.7 sample\_tile\_pyramid (Informative)

**Table 29 EXAMPLE: tiles table Create Table SQL**

```
CREATE TABLE sample_tile_pyramid (
 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 30 EXAMPLE: tiles table Insert Statement**

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

### C.8 gpkg\_data\_columns

**Table 31 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_name TEXT,
 CONSTRAINT pk_gdc PRIMARY KEY (table_name, column_name),
 CONSTRAINT fk_gdc_tn FOREIGN KEY (table_name)
 REFERENCES gpkg_contents(table_name))
```

## C.9 gpkg\_data\_column\_constraints

**Table 32 gpkg\_data\_column\_constraints Table Definition SQL**

```
CREATE TABLE gpkg_data_column_constraints (
 constraint_name TEXT NOT NULL,
 constraint_type TEXT NOT NULL, // 'range' | 'enum' | 'glob'
 value TEXT,
 min NUMERIC,
 minIsInclusive BOOLEAN, /* 0 = false, 1 = true */
 max NUMERIC,
 maxIsInclusive BOOLEAN, /* 0 = false, 1 = true */
 Description TEXT,
 CONSTRAINT gdcc_ntv UNIQUE
 (constraint_name, constraint_type, value)
)
```

## C.10 gpkg\_metadata

**Table 33 gpkg\_metadata Table Definition SQL**

```
CREATE TABLE gpkg_metadata (
 id INTEGER CONSTRAINT m_pk PRIMARY KEY ASC
 AUTOINCREMENT NOT NULL UNIQUE,
 md_scope TEXT NOT NULL DEFAULT 'dataset',
 metadata_standard_uri TEXT NOT NULL,
 mime_type TEXT NOT NULL DEFAULT 'text/xml',
 metadata TEXT NOT NULL
)
```

## C.11 gpkg\_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 DATETIME 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)
```

## C.12 gpkg\_extensions

**Table 36 gpkg\_extensions Table Definition SQL**

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

## Annex D Trigger Definition SQL (Informative)

### D.1 gpkg\_tile\_matrix

Table 37 gpkg\_tile\_matrix Trigger Definition SQL

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

CREATE TRIGGER 'gpkg_tile_matrix_zoom_level_update'
BEFORE UPDATE OF zoom_level ON 'gpkg_tile_matrix'
FOR EACH ROW BEGIN
SELECT RAISE(ABORT, 'update on table 'gpkg_tile_matrix' violates
constraint: zoom_level cannot be less than 0')
WHERE (NEW.zoom_level < 0);
END

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

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

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

CREATE TRIGGER 'gpkg_tile_matrix_matrix_height_update'
BEFORE UPDATE OF matrix_height ON 'gpkg_tile_matrix'
FOR EACH ROW BEGIN
SELECT RAISE(ABORT, 'update on table 'gpkg_tile_matrix' violates
constraint: matrix height cannot be less than 1')
```

```

WHERE (NEW.matrix_height < 1);
END

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

CREATE TRIGGER 'gpkg_tile_matrix_pixel_x_size_update'
BEFORE UPDATE OF pixel_x_size ON 'gpkg_tile_matrix'
FOR EACH ROW BEGIN
SELECT RAISE(ABORT, 'update on table 'gpkg_tile_matrix' violates
constraint: pixel_x_size must be greater than 0')
WHERE NOT (NEW.pixel_x_size > 0);
END

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

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

```

## D.2 metadata

**Table 38 metadata Trigger Definition SQL**

```

CREATE TRIGGER 'gpkg_metadata_md_scope_insert'
BEFORE INSERT ON 'gpkg_metadata'
FOR EACH ROW BEGIN
SELECT RAISE(ABORT, '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(ABORT, '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 |
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
```

### D.3 metadata\_reference

**Table 39 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(ABORT, '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(ABORT, 'update 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_column_name_insert'
BEFORE INSERT ON 'gpkg_metadata_reference'
FOR EACH ROW BEGIN
SELECT RAISE(ABORT, '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(ABORT, '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'
BEFORE UPDATE OF column_name ON 'gpkg_metadata_reference'
FOR EACH ROW BEGIN
SELECT RAISE(ABORT, '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_name IS NOT NULL);
SELECT RAISE(ABORT, '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(ABORT, '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(ABORT, '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(ABORT, '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(ABORT, '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

CREATE TRIGGER 'gpkg_metadata_reference_timestamp_insert'
BEFORE INSERT ON 'gpkg_metadata_reference'
FOR EACH ROW BEGIN
SELECT RAISE(ABORT, '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(ABORT, '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

```

#### D.4 sample\_feature\_table

**Table 40 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(ABORT, '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 (ABORT, '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.

### D.5 sample\_tile\_pyramid

**Table 41 EXAMPLE: tiles table Trigger Definition SQL**

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

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

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

CREATE TRIGGER "sample_tile_pyramid_tile_column_update"
BEFORE UPDATE OF tile_column ON "sample_tile_pyramid"
FOR EACH ROW BEGIN
SELECT RAISE(ABORT, 'update on table 'sample_tile_pyramid''
violates constraint: tile_column cannot be < 0')
WHERE (NEW.tile_column < 0) ;
```

```

SELECT RAISE(ABORT, 'update on table 'sample_tile_pyramid'
violates constraint: tile_column must by < matrix_width specified
for table and zoom level in gpkg_tile_matrix')
WHERE NOT (NEW.tile_column < (SELECT matrix_width FROM
gpkg_tile_matrix WHERE t_table_name = 'sample_tile_pyramid' AND
zoom_level = NEW.zoom_level));
END

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

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

```

## Annex E Geometry Types (Normative)

**Table 42 Geometry Type Codes (Core)**

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

**Table 43 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.

SURFACE subtype is CURVEPOLYGON.

CURVEPOLYGON subtype is POLYGON.

GEOMCOLLECTION subtypes are MULTIPOINT, MULTICURVE and MULTISURFACE.

MULTICURVE subtype is MULTILINESTRING.

MULTISURFACE subtype is MULTIPOLYGON

## Annex F Tiles Zoom Times Two Example (Informative)

Table 44 Zoom Times Two Example

| table_name | zoom_level | matrix_width | matrix_height | tile_width | tile_height | pixel_x_size | pixel_y_size |
|------------|------------|--------------|---------------|------------|-------------|--------------|--------------|
| "MyTiles"  | 0          | 8            | 8             | 512        | 512         | 69237.2      | 68412.1      |
| "MyTiles"  | 1          | 16           | 16            | 512        | 512         | 34618.6      | 34206.0      |
| "MyTiles"  | 2          | 32           | 32            | 512        | 512         | 17309.3      | 17103.0      |
| "MyTiles"  | 3          | 64           | 64            | 512        | 512         | 8654.64      | 8654.64      |
| "MyTiles"  | 4          | 128          | 128           | 512        | 512         | 4327.32      | 4275.75      |
| "MyTiles"  | 5          | 256          | 256           | 512        | 512         | 2163.66      | 2137.87      |
| "MyTiles"  | 6          | 512          | 512           | 512        | 512         | 1081.83      | 1068.93      |
| "MyTiles"  | 7          | 1024         | 1024          | 512        | 512         | 540.915      | 534.469      |
| "MyTiles"  | 8          | 2048         | 2048          | 512        | 512         | 270.457      | 267.234      |

## Annex G Hierarchical Metadata Example (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,
 metadata TEXT NOT NULL
)
```

| id | md_scope      | metadata_standard_uri                                                           | metadata |
|----|---------------|---------------------------------------------------------------------------------|----------|
| 0  | undefined     | <a href="http://www.isotc211.org/2005/gmd">http://www.isotc211.org/2005/gmd</a> | TEXT     |
| 3  | series        | <a href="http://www.isotc211.org/2005/gmd">http://www.isotc211.org/2005/gmd</a> | TEXT     |
| 4  | dataset       | <a href="http://www.isotc211.org/2005/gmd">http://www.isotc211.org/2005/gmd</a> | TEXT     |
| 5  | featureType   | <a href="http://www.isotc211.org/2005/gmd">http://www.isotc211.org/2005/gmd</a> | TEXT     |
| 6  | feature       | <a href="http://www.isotc211.org/2005/gmd">http://www.isotc211.org/2005/gmd</a> | TEXT     |
| 7  | attributeType | <a href="http://www.isotc211.org/2005/gmd">http://www.isotc211.org/2005/gmd</a> | TEXT     |
| 8  | attribute     | <a href="http://www.isotc211.org/2005/gmd">http://www.isotc211.org/2005/gmd</a> | TEXT     |

and this reference table definition:

```
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)
)
```

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')),
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 data 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 */
-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    | <a href="http://www.isotc211.org/2005/gmd">http://www.isotc211.org/2005/gmd</a> | TEXT     |
| 1  | fieldSession | <a href="http://www.isotc211.org/2005/gmd">http://www.isotc211.org/2005/gmd</a> | TEXT     |
| 10 | featureType  | <a href="http://www.isotc211.org/2005/gmd">http://www.isotc211.org/2005/gmd</a> | TEXT     |
| 11 | feature      | <a href="http://www.isotc211.org/2005/gmd">http://www.isotc211.org/2005/gmd</a> | TEXT     |
| 12 | attribute    | <a href="http://www.isotc211.org/2005/gmd">http://www.isotc211.org/2005/gmd</a> | TEXT     |
| 13 | attribute    | <a href="http://www.isotc211.org/2005/gmd">http://www.isotc211.org/2005/gmd</a> | TEXT     |
| 14 | feature      | <a href="http://www.isotc211.org/2005/gmd">http://www.isotc211.org/2005/gmd</a> | TEXT     |
| 15 | attribute    | <a href="http://www.isotc211.org/2005/gmd">http://www.isotc211.org/2005/gmd</a> | TEXT     |
| 16 | attribute    | <a href="http://www.isotc211.org/2005/gmd">http://www.isotc211.org/2005/gmd</a> | TEXT     |
| 17 | feature      | <a href="http://www.isotc211.org/2005/gmd">http://www.isotc211.org/2005/gmd</a> | TEXT     |
| 18 | attribute    | <a href="http://www.isotc211.org/2005/gmd">http://www.isotc211.org/2005/gmd</a> | TEXT     |
| 19 | attribute    | <a href="http://www.isotc211.org/2005/gmd">http://www.isotc211.org/2005/gmd</a> | TEXT     |

(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 */
)
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')),
 12, /* md_file_id */
 1 /* md_parent_id */
)
```

```

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 */
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 | <a href="http://www.isotc211.org/2005/gmd">http://www.isotc211.org/2005/gmd</a> | TEXT     |

|    |                   |                                                                                 |      |
|----|-------------------|---------------------------------------------------------------------------------|------|
| 1  | fieldSession      | <a href="http://www.isotc211.org/2005/gmd">http://www.isotc211.org/2005/gmd</a> | TEXT |
| 2  | collectionSession | <a href="http://www.isotc211.org/2005/gmd">http://www.isotc211.org/2005/gmd</a> | TEXT |
| 3  | series            | <a href="http://www.isotc211.org/2005/gmd">http://www.isotc211.org/2005/gmd</a> | TEXT |
| 4  | dataset           | <a href="http://www.isotc211.org/2005/gmd">http://www.isotc211.org/2005/gmd</a> | TEXT |
| 5  | featureType       | <a href="http://www.isotc211.org/2005/gmd">http://www.isotc211.org/2005/gmd</a> | TEXT |
| 6  | feature           | <a href="http://www.isotc211.org/2005/gmd">http://www.isotc211.org/2005/gmd</a> | TEXT |
| 7  | attributeType     | <a href="http://www.isotc211.org/2005/gmd">http://www.isotc211.org/2005/gmd</a> | TEXT |
| 8  | attribute         | <a href="http://www.isotc211.org/2005/gmd">http://www.isotc211.org/2005/gmd</a> | TEXT |
| 10 | featureType       | <a href="http://www.isotc211.org/2005/gmd">http://www.isotc211.org/2005/gmd</a> | TEXT |
| 11 | feature           | <a href="http://www.isotc211.org/2005/gmd">http://www.isotc211.org/2005/gmd</a> | TEXT |
| 12 | attribute         | <a href="http://www.isotc211.org/2005/gmd">http://www.isotc211.org/2005/gmd</a> | TEXT |
| 13 | attribute         | <a href="http://www.isotc211.org/2005/gmd">http://www.isotc211.org/2005/gmd</a> | TEXT |
| 14 | feature           | <a href="http://www.isotc211.org/2005/gmd">http://www.isotc211.org/2005/gmd</a> | TEXT |
| 15 | attribute         | <a href="http://www.isotc211.org/2005/gmd">http://www.isotc211.org/2005/gmd</a> | TEXT |
| 16 | attribute         | <a href="http://www.isotc211.org/2005/gmd">http://www.isotc211.org/2005/gmd</a> | TEXT |
| 17 | feature           | <a href="http://www.isotc211.org/2005/gmd">http://www.isotc211.org/2005/gmd</a> | TEXT |
| 18 | attribute         | <a href="http://www.isotc211.org/2005/gmd">http://www.isotc211.org/2005/gmd</a> | TEXT |
| 19 | attribute         | <a href="http://www.isotc211.org/2005/gmd">http://www.isotc211.org/2005/gmd</a> | TEXT |

## 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            |

## **Annex H 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.

- |                                                                                                                                                                                                                                                                                                |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"><li>• Rational Polynomial Coefficient</li><li>• Photometric Interpretation</li><li>• No Data Value</li><li>• Compression Quality Factor</li><li>• Georectification</li><li>• NIIRS</li><li>• Min X</li><li>• Min Y</li><li>• Max X</li><li>• Max Y</li></ul> |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Annex I GeoPackage Extension Template (Normative)<sup>1</sup>

### *Extension Title*

Title of the Extension

### *Introduction*

Description of extension

### *Extension Author*

Author of extension, author\_name.

### *Extension Name or Template*

Name of the extension or definition of the template to create the name of extensions that should be used in gpkg\_extensions

### *Extension Type*

“Extension of **Existing** Requirement in Clause(s) XXX” or

“**New** Requirement Dependent on Clause(s) YYY”

### *Applicability*

Tables and/or columns on which this extension may be applied

### *Scope*

Read-write or write-only with clarification if necessary

### *Requirements*

Definition of extension and interdependencies with other extensions if any.

### *GeoPackage*

Definition of extension data or MIME type(s)

Definition of extension tables or table templates

Definition of triggers or trigger templates

### *GeoPackage SQLite Configuration*

---

<sup>1</sup> This template was used to create the following annexes. The form and outline heading numbering are irrelevant. ASCII output document format is preferred for inclusion in the gpkg\_extensions definition column.

## Definition of SQLite configuration settings

| Setting<br>compile<br>or<br>runtime | Option | Shall<br>/ Not<br>(Value) | Discussion |
|-------------------------------------|--------|---------------------------|------------|
|                                     |        |                           |            |

*GeoPackage SQLite Extension*

## Definition of SQL functions

| SQL Function             | Description      | Use |
|--------------------------|------------------|-----|
| foo(bar, baz) : datatype | Returns r when w |     |

## Annex J GeoPackage Geometry Types Extension Template (Normative)

### *Extension Title*

GeoPackage Non-Linear Geometry Types

### *Introduction*

Clause 2.1.4 of the GeoPackage Version 1 Encoding Standard specifies support for the Geometry, Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon, and GeomCollection geometry types in the GeoPackageBinary geometry encoding format specified in clause 2.1.3. This extension specifies support for the additional CircularString, CompoundCurve, CurvePolygon, MultiCurve, MultiSurface, Curve, and Surface geometry types in the GeoPackage Binary geometry encoding format using the codes listed in Annex D.

### *Extension Author*

GeoPackage SWG, author\_name “gpkg”

### *Extension Name or Template*

Extension names are constructed from the gpkg\_geom\_<gname> template where <gname> is the uppercase name of the extension geometry type in Annex D Table 52?

### *Extension Type*

Extension of **Existing** Requirement in clause 2.1.4.

### *Applicability*

This extension applies to any column specified in the gpkg\_geometry\_columns table.

### *Scope*

Read-write

### *Requirements*

#### *GeoPackage*

The GeoPackageBinary geometry encoding format specified in clause 2.1.3 SHALL be used to encode non-linear geometry types using the type codes in Annex E Table 43.

#### *GeoPackage SQLite Configuration*

None

#### *GeoPackage SQLite Extension*

SQL functions that operate on GeoPackageBinary geometries as specified in other extensions SHALL operate correctly on the non-linear geometries specified in this extension.

## Annex K User-Defined Geometry Types Extension Template (Normative)

### *Extension Title*

User Defined Geometry Types Extension of GeoPackageBinary Geometry Encoding

### *Introduction*

This extension specifies a standard way to implement user defined extensions of the GeoPackageBinary geometry encoding format to encode geometry types not specified in clauses 2.1.4 and 3.1.1 and listed in Annex D. It is intended to be a bridge to enable use of geometry types like EllipticalCurve in Extended GeoPackages until standard encodings of such types are developed and published for the Well Known Binary (WKB) format.

### *Extension Author*

Name of implementer, author\_name NOT “gpkg”.

### *Extension Name or Template*

Extension names are constructed from the <author\_name>\_geom\_<gname> template where <gname> is the uppercase name of an extension geometry type NOT in Annex E.

### *Extension of Existing or New Requirement*

Extension of **Existing** Requirements in clauses 2.1.3, 2.1.4, 2.1.5 and 3.1.1.

### *Applicability*

This extension applies to any column specified in the gpkg\_geometry\_columns table.

### *Scope*

Read-write

### *Requirements*

This extension specifies use of an ExtendedGeoPackageBinary encoding format for geometry types not listed in Annex E, and use of the extension name in uppercase for the geometry\_type\_name column value in the gpkg\_geometry\_columns table.

### *GeoPackage*

One of the reserved bits in the GeoPackageBinary header is used to indicate the presence of the ExtendedGeopackageBinary encoding format. In the extension case a four byte sequence follows the GPB header to disambiguate various extensions. This extension\_code SHOULD identify the implementer of the extension and/or the particular geometry type extension, and

SHOULD be unique. The actual extension geometry body is not specified, but SHALL be described in the extension document.

```
ExtendedGeoPackageBinary {
 GeoPackageBinaryHeader header; // The X bit in the header flags field
 // must be set to 1.
 byte[4] extension_code; // To indicate different extensions or vendors.
 // 0x47504B47 (GPKG in ASCII) is reserved.
 byte[] extension_specific; // Extension specific contents
}
```

### *GeoPackage SQLite Configuration*

None.

### *GeoPackage SQLite Extension*

SQL functions that operate on GeoPackageBinary geometries as specified in other extensions SHALL operate correctly on user-defined geometry types encoded in the ExtendedGeopackageBinary format as specified in this extension.

## Annex L Rtree Spatial Index Extension (Normative)

### *Extension Title*

Rtree Spatial Indexes

### *Introduction*

The rtree index extension provides a means to encode an rtree index for geometry values in a GeoPackage. An RTree index provides 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.

### *Extension Author*

GeoPackage SWG, author\_name “gpkg”.

### *Extension Name or Template*

“gpkg\_rtree\_index”

### *Extension Type*

**New** Requirement dependent on clauses 2.1.3 and 3.1.2.

### *Applicability*

This extension applies to any column specified in the gpkg\_geometry\_columns table.

### *Scope*

Write-only, because it does not change the result of reads, although it may improve their performance.

### *Requirements*

This extension uses the rtree implementation provided by the SQLite R\*Tree Module extension documented at <http://www.sqlite.org/rtree.html>.

### *GeoPackage*

The tables below contain SQL templates with variables. Replace the following template variables with the specified values to create the required SQL statements:

<t>: The name of the feature table containing the geometry column

<c>: The name of the geometry column in <t> that is being indexed

<i>: The name of the integer primary key column in <t> as specified in Req 25

### *Create Virtual Table*

Rtree spatial indexes on geometry columns 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:

```
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.

### *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

```
INSERT OR REPLACE INTO rtree_<t>_<c>
SELECT <i>, 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 and <i> is replaced with the name of the feature table integer primary key column.

### *Define Triggers to Maintain Spatial Index Values*

For each spatial index in a GeoPackage, 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:

```
/* 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.<i>,
 ST_MinX(NEW.<c>), ST_MaxX(NEW.<c>),
 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.<i> = NEW.<i> AND
 (NEW.<c> NOTNULL AND NOT ST_IsEmpty(NEW.<c>))
BEGIN
 INSERT OR REPLACE INTO rtree_<t>_<c> VALUES (
 NEW.<i>,
 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.<i> = NEW.<i> AND
 (NEW.<c> ISNULL OR ST_IsEmpty(NEW.<c>))
BEGIN
 DELETE FROM rtree_<t>_<c> WHERE id = OLD.<i>;
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.<i> != NEW.<i> AND
 (NEW.<c> NOTNULL AND NOT ST_IsEmpty(NEW.<c>))
BEGIN
 DELETE FROM rtree_<t>_<c> WHERE id = OLD.<i>;
 INSERT OR REPLACE INTO rtree_<t>_<c> VALUES (
 NEW.<i>,
 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.<i> != NEW.<i> AND
 (NEW.<c> ISNULL OR ST_IsEmpty(NEW.<c>))
BEGIN
 DELETE FROM rtree_<t>_<c> WHERE id IN (OLD.<i>, NEW.<i>);
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.<id>;
END;

```

where <t> and <c> are replaced with the names of the feature table and geometry column being indexed and <i> is replaced with the name of the feature table integer primary key column.

### *GeoPackage SQLite Configuration*

| Setting | Option                | Shall / Not (Value) | Discussion                                                                |
|---------|-----------------------|---------------------|---------------------------------------------------------------------------|
| compile | SQLITE_ENABLE_RTREE   | Shall               | Rtrees are used for GeoPackage Spatial Indexes                            |
| compile | SQLITE_RTREE_INT_ONLY | Not                 | Rtrees with floating point values are used for GeoPackage Spatial Indexes |

### *GeoPackage SQLite Extension*

| SQL Function                           | Description                                                                           | Use                                                               |
|----------------------------------------|---------------------------------------------------------------------------------------|-------------------------------------------------------------------|
| ST_IsEmpty(geom. Geometry):<br>INTEGER | 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_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 Envelope of a Geometry                    | Update the spatial index on a geometry column in a feature table. |
| 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. |

The SQL functions on geometries in this SQLite Extension SHALL operate correctly on extended geometry types specified by Annex J and/or Annex K when those extensions are also implemented.

## Annex M Geometry Type Triggers Extension (Normative)

### *Extension Title*

Geometry Type Triggers

### *Introduction*

Geometry type triggers prevent the storage of geometries of types that are not assignable from the geometry types specified in the `gpkg_geometry_columns` table in the geometry columns of the specified tables.

### *Extension Author*

GeoPackage SWG, author\_name “gpkg”.

### *Extension Name or Template*

“gpkg\_geometry\_type\_trigger”

### *Extension Type*

**New** Requirement dependent on clauses 2.1.3 and 3.1.2.

### *Applicability*

This extension applies to any column specified in the `gpkg_geometry_columns` table.

### *Scope*

Write-only

### *Requirements*

The `<t>` and `<c>` template parameters in the geometry type trigger definition SQL template in Table xx below are to be replaced with the names of the feature table and geometry column being inserted or updated.

### *GeoPackage*

```
CREATE TRIGGER fgti_<t>_<c> BEFORE INSERT ON '<t>'
FOR EACH ROW
BEGIN
 SELECT RAISE (ABORT, '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

CREATE TRIGGER fgtu_<t>_<c> BEFORE UPDATE OF '<c>' ON '<t>'
FOR EACH ROW
BEGIN
SELECT RAISE (ABORT,
'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

```

### *GeoPackage SQLite Configuration*

**None.**

### *GeoPackage SQLite Extension*

| SQL Function                                                               | Description                                                                                                                | Use                                                                                                 |
|----------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------|
| ST_GeometryType(geom. Geometry) : TEXT                                     | Returns the WKB geometry type name of a Geometry                                                                           | Check that the geometry type matches what's specified in gpkg_geometry_columns.geometry_type_name   |
| 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. |

The SQL functions on geometries in this SQLite Extension SHALL operate correctly on extended geometry types specified by Annex J and/or Annex K when those extensions are also implemented.

## Annex N Geometry SRS\_ID Triggers Extension (Normative)

### *Extension Title*

Geometry SRS\_ID Triggers

### *Introduction*

Geometry SRS\_ID triggers prevent the storage of geometries with spatial reference system identifiers that are not specified in the `gpkg_geometry_columns` table in the geometry columns of the specified tables.

### *Extension Author*

GeoPackage SWG, author\_name “gpkg”.

### *Extension Name or Template*

“gpkg\_srs\_id\_trigger”

### *Extension Type*

**New** Requirement dependent on clauses 2.1.3 and 3.1.2.

### *Applicability*

This extension applies to any column specified in the `gpkg_geometry_columns` table.

### *Scope*

Write-only

### *Requirements*

The `<t>` and `<c>` template parameters in the SRS\_ID trigger definition SQL template in Table xx below are to be replaced with the names of the feature table and geometry column being inserted or updated

### *GeoPackage*

```
CREATE TRIGGER fgssi_<t> _<c> BEFORE INSERT ON '<t>'
FOR EACH ROW
BEGIN
 SELECT RAISE (ABORT, '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 (ABORT,
'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

```

### *GeoPackage SQLite Configuration*

**None.**

### *GeoPackage SQLite Extensions*

| 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 gpkg_geometry_columns.srid |

The SQL function on geometries in this SQLite Extension SHALL operate correctly on extended geometry types specified by Annex J and/or Annex K when those extensions are also implemented.

## Annex O Zoom Other Intervals Extension (Normative)

### *Extension Title*

Zoom Other Intervals

### *Introduction*

In a GeoPackage, zoom levels are integers in sequence from 0 to n that identify tile matrix layers in a tile matrix set that contain tiles of decreasing spatial extent and finer spatial resolution. Adjacent zoom levels immediately precede or follow each other and differ by a value of 1. Pixel sizes are real numbers in the terrain units of the spatial reference system of a tile image specifying the dimensions of the real world area represented by one pixel. Pixel sizes MAY vary by a constant factor or by different factors or intervals between some or all adjacent zoom levels in a tile matrix set. In the commonly used "zoom times two" convention, pixel sizes vary by a factor of 2 between all adjacent zoom levels, as shown in the example in Annex F.

This extension enables use of "zoom other intervals" conventions with different factors or irregular intervals with pixel sizes chosen for intuitive cartographic representation of raster data, or to coincide with the original pixel size of commonly used global image products. See WMTS [16] Annex E for additional examples of both conventions.

### *Extension Author*

GeoPackage SWG, author\_name "gpkg".

### *Extension Name or Template*

"gpkg\_zoom\_other

### *Extension Type*

Extension of **Existing** Requirement in clause 2.2.3.

### *Applicability*

This extension applies to any table listed in the gpkg\_contents table with a data\_type of "tiles".

### *Scope*

Read-write

### *Requirements*

#### *GeoPackage*

The pixel\_x\_size and / or pixel\_y\_size column values in the gpkg\_tile\_matrix table vary by irregular intervals or by regular intervals other than a factor of two (the default) between adjacent zoom levels for a particular tile matrix set pyramid table.

#### *GeoPackage SQLite Configuration*

None

#### *GeoPackage SQLite Extension*

None

## Annex P Tile Encoding WebP Extension

### *Extension Title*

Tile Encoding WebP

### *Introduction*

PNG and JPEG are the default MIME types for encoding images in tile pyramid user data tables. This extension allows the use of image/x-webp as an additional encoding type.

### *Extension Author*

GeoPackage SWG, author\_name “gpkg”.

### *Extension Name or Template*

“gpkg\_webp”

### *Extension Type*

Extension of **Existing** Requirement in clauses 2.2.4 and 2.2.5.

### *Applicability*

This extension applies to any table listed in the gpkg\_contents table with a data\_type of “tiles”.

### *Scope*

Read-write

### *Requirements*

#### *GeoPackage*

The MIME type of values of the tile\_data column in tile pyramid user data tables SHALL be image/x-webp.

#### *GeoPackage SQLite Configuration*

None

#### *GeoPackage SQLite Extension*

None

## Annex Q 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/IEC 9075:1992 Information Technology - Database Language SQL SQL92)
- [2] ISO/IEC 9075-1:2011 Information Technology - Database Language SQL - Part 1: Framework
- [3] ISO/IEC 9075-2:2011 Information Technology - Database Language SQL - Part 2: Foundation
- [4] ISO/IEC 9075-3:2008 Information Technology - Database Language SQL - Part 3: Call-Level Interface (SQL/CLI)
- [5] SQLite (all parts) <http://www.sqlite.org/> (online) <http://www.sqlite.org/sqlite-doc-3071300.zip> (offline)
- [6] <http://sqlite.org/fileformat2.html>
- [7] <http://www.sqlite.org/formatchg.html>
- [8] <http://www.sqlite.org/download.html>
- [9] 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](http://portal.opengeospatial.org/files/?artifact_id=25355) (also ISO/TC211 19125 Part 1)
- [10] 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](http://portal.opengeospatial.org/files/?artifact_id=25354) (also ISO/TC211 19125 Part 2)
- [11] 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](http://portal.opengeospatial.org/files/?artifact_id=829)
- [12] ISO/IEC 13249-3:2011 Information technology — SQL Multimedia and Application Packages - Part 3: Spatial (SQL/MM)
- [13] <http://www.epsg.org/Geodetic.html>
- [14] <http://www.epsg-registry.org/>
- [15] MIL\_STD\_2401 DoD World Geodetic System 84 (WGS84), 11 January 1994
- [16] OGC 07-057r7 OpenGIS® Web Map Tile Service Implementation Standard version 1.0.0 2010-04-06 (WMTS) [http://portal.opengeospatial.org/files/?artifact\\_id=35326](http://portal.opengeospatial.org/files/?artifact_id=35326)
- [17] ITU-T Recommendation T.81 (09/92) with Corrigendum (JPEG)
- [18] JPEG File Interchange Format Version 1.02, September 1, 1992  
<http://www.jpeg.org/public/jfif.pdf>

- [19] IETF RFC 2046 Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types  
<http://www.ietf.org/rfc/rfc2046.txt>
- [20] Portable Network Graphics <http://libpng.org/pub/png/>
- [21] MIME Media Types <http://www.iana.org/assignments/media-types/index.html>
- [22] WebP <https://developers.google.com/speed/webp/>
- [23] IETF RFC 3986 Uniform Resource Identifier (URI): Generic Syntax  
<http://www.ietf.org/rfc/rfc3986.txt>
- [24] W3C Recommendation 26 November 2008 Extensible Markup Language (XML) 1.0 (Fifth Edition) <http://www.w3.org/TR/xml/>
- [25] W3C Recommendation 28 October 2004 XML Schema Part 0: Primer Second Edition  
<http://www.w3.org/TR/xmlschema-0/>
- [26] W3C Recommendation 28 October 2004 XML Schema Part 1: Structures Second Edition  
<http://www.w3.org/TR/xmlschema-1/>
- [27] W3C Recommendation 28 October 2004 XML Schema Part 2: Datatypes Second Edition  
<http://www.w3.org/TR/xmlschema-2/>
- [28] ISO 19115 Geographic information -- Metadata, 8 May 2003, with Technical Corrigendum 1, 5 July 2006
- [29] ISO 8601 Representation of dates and times  
[http://www.iso.org/iso/catalogue\\_detail?csnumber=40874](http://www.iso.org/iso/catalogue_detail?csnumber=40874)
- [30] OGC® 10-100r3 Geography Markup Language (GML) simple features profile (with technical note) [http://portal.opengeospatial.org/files/?artifact\\_id=42729](http://portal.opengeospatial.org/files/?artifact_id=42729)
- [31] SQLite R\*Tree Module <http://www.sqlite.org/rtree.html>

## Annex R 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://en.wikipedia.org/wiki/ASCII>
- [B5] [http://www.sqlite.org/lang\\_createtable.html#rowid](http://www.sqlite.org/lang_createtable.html#rowid)
- [B6] ISO 19115-2 Geographic information -- Metadata - Part 2: Metadata for imagery and gridded data
- [B7] ISO 19139: Geographic information -- Metadata -- XML schema implementation
- [B8] Dublin Core Metadata Initiative <http://dublincore.org/> [IETF RFC 5013](http://dublincore.org/documents/2012/06/07/dcmi-terms/)
- [B9] ISO 15836:2009  
[http://www.iso.org/iso/home/store/catalogue\\_tc/catalogue\\_detail.htm?csnumber=52142](http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=52142)
- [B10] Content Standard for Digital Geospatial Metadata (CSDGM)
- [B11] <http://www.fgdc.gov/standards/projects/FGDC-standards-projects/metadata/base-metadata/index.html>
- [B12] Department of Defense Discovery Metadata Specification (DDMS)  
<http://metadata.ces.mil/mdr/irs/DDMS/>
- [B13] NMF [NGA.STND.0012\\_2.0](#) / NMIS [NGA.STND.0018\\_1.0](#)
- [B14] Unified Modeling Language (UML) <http://www.uml.org/>
- [B15] XML for Metadata Interchange (XMI) <http://www.omg.org/spec/XMI/>
- [B16] IDEF1x Data Modeling Method <http://www.idef.com/IDEF1x.htm>
- [B17] Geography Markup Language (GML) [ISO 19136:2007](#)
- [B18] ISO 19110 Geographic information – Methodology for feature cataloguing
- [B19] RDF Vocabulary Description Language 1.0: RDF Schema  
<http://www.w3.org/TR/rdf-schema/>
- [B20] Web Ontology Language (OWL) <http://www.w3.org/TR/2009/REC-owl2-xml-serialization-20091027/>
- [B21] Simple Knowledge Organization System (SKOS) <http://www.w3.org/TR/skos-reference/>
- [B22] STANAG 7074 Digital Geographic Information Exchange Standard (DIGEST) - AGeoP-3A, edition 1, 19 October 1994  
[http://www.dgiwg.org/dgiwg/hm/documents/historical\\_documents.htm](http://www.dgiwg.org/dgiwg/hm/documents/historical_documents.htm)
- [B23] ISO 19109 Geographic information - Rules for application schema
- [B24] <http://www.sqlite.org/changes.html>

[B25] <http://sqlite.org/src4/doc/trunk/www/design.wiki>