Universal spatial DataBase eXtension (UDBX)

オープンデータフォーマットホワイトペーパー

UDBX Open Data Format WhitePaper

Draft

日本スーパーマップ株式会社

2020年10月

目　次

1　概要

UDBX（(Universal Spatial Database Extension)は、SuperMapが提案したファイルタイプのオープンデータフォーマットです。空間データの効率的な保存と管理をサポートし、空間データの共有と交換のためのオープンで便利なソリューションを提供します。

UDBXには次のような主な特長があります。

●全空間データサポート：2、3次元統合とベクタ/ラスタ統合のデータストレージと管理を含む。

●シングルファイルストレージ：データのコピーと配布を容易にするために、単一のファイルストレージを使用します。

●同時実行サポート：1つのUDBXデータは、複数のユーザーが同時にアクセスできます。

●クロスプラットフォームのサポート：デスクトップ、サーバー、モバイル端末での効率的な読み書きをサポートします。

●大量データのサポート：1つのUDBXファイルは、超大容量のデータストレージをサポートします。

●Unicodeサポート：複数の言語を格納するためのUnicodeエンコーディングをサポートします。

1.1 準拠

本書に記載されている用語または略語は、次の規則に準拠します。

●OGC：オープン地理空間情報コンソーシアム(OpenGeospatialConsortium)

●OGCシンプルオブジェクトモデル仕様：OGCSimplefeatureaccess-Part1：Commonarchitecture仕様に規定さ●れている単純なオブジェクトモデル

●SQLite：オープンソースの軽量版リレーショナルデータベース

●SpatiaLite：SQLiteベースの空間拡張エンジンに基づいて、OGC単純オブジェクトモデル仕様に準拠した空間データの格納および管理

●proj.4：OSGeoのオープンソースGISツール。マップの座標系表現と変換

●WKT：OGCのWellKnownText

●データセット：同じ座標系を持つ同じ種類のデータで構成されるデータコレクション

●ベクタデータセット：CADデータセットを除き、同じGeometryタイプ、同じ空間参照系、同じフィールド構造を持つベクタフィーチャコレクション

●ラスタデータセット：長さと幅の範囲を持った規則的な空間配列データセット。配列データは同じ空間参照系を持ち、空間位置に対応した同類の属性値、例えばは温度、標高値などを有するもの

●データソース：さまざまな種類のデータセットで構成されるデータセットのコレクション

現在、UDBXが対応しているバージョンはSpatiaLite 4.3、SQLite 3.17.0となっています。

フィールド/データタイプについて

●本書のシステムテーブル(セクション2を参照)とデータテーブル(セクション3を参照)の表“の[フィールドの種類]はSQLiteです。

●INTEGER型がデフォルトで32ビット整数であるフィールド型。

●UDBXによって提供されるデータセットフィールド情報は、SQLiteフィールドの種類に基づいて拡張されます。2.2.2.2セクション;

●オブジェクトのバイナリストリームストアは、基本的なデータ型で記述され、各型の意味、値の範囲、占有バイト数などは、セクション4.1.1を参照してください。

1.2 全体構造

UDBXはSQLiteデータベースでデータを保存して、ファイル拡張名は"udbx"です。ストレージルールは、SpatiaLiteストレージとSuperMapカスタムストレージ2つのタイプがあります。SpatiaLiteストレージはポイント、ライン、ポリゴンなどの基本的なベクタタイプに対応して、SuperMapカスタムストレージ形式はその他のデータタイプに対応します。

UDBXのテーブルは、システムテーブルとデータテーブルの2つのカテゴリに区分されます。システムテーブルはUDBXのデータセットを管理するために使用されるため、システムテーブルはSpatiaLiteのシステムテーブルとSuperMapカスタムシステムテーブルと区別されます。UDBXのテーブル構造の関係を図1に示します。

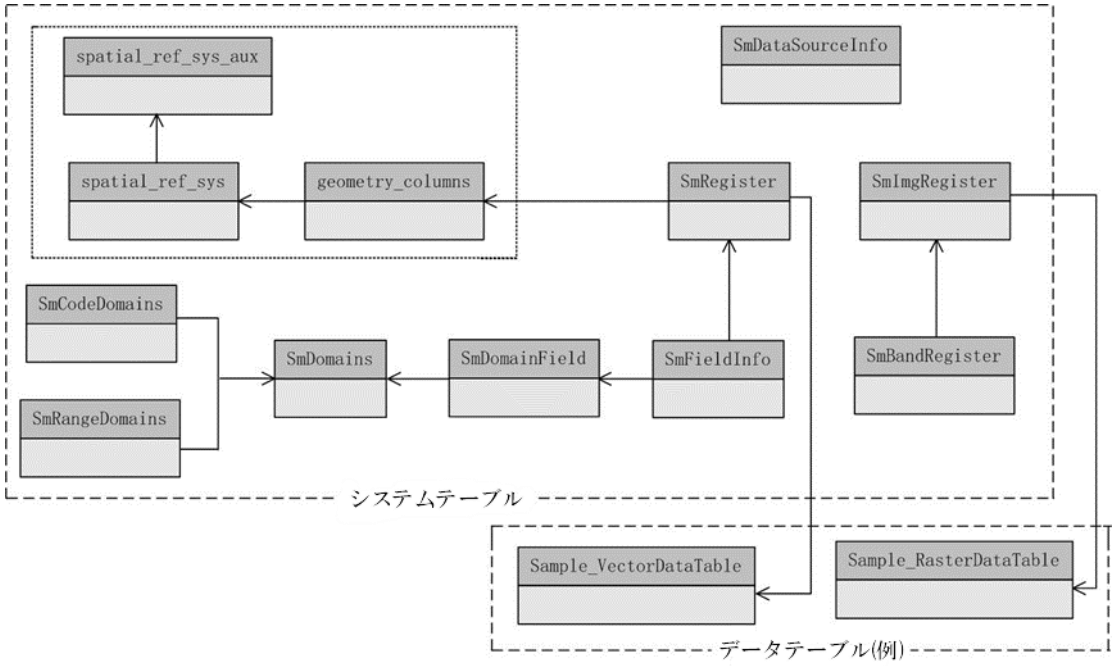

図1 UDBXテーブル構造図

1.3 **データセットのタイプ**

一つのudbxファイルは、複数のデータセットを格納する1つのデータソースであります。データセットには、ベクタデータセットとラスタデータセットの2つのカテゴリがあります。ベクタデータセットには、同じ座標系と属性情報を持つ空間オブジェクトが格納され、ラスタデータセットは連続したフィールドデータをブロック別に格納および整理します。データセットの種類を表1に示します。

表1 データセットの種類の説明

データセットタイプ	列挙値	説　　明
Tabular	0	幾何オブジェクトを格納しないテーブル
Point	1	2Dポイントデータセット
PointZ	101	3Dポイントデータセット
Lin.	3	2Dラインデータセット
LineZ	103	3Dラインデータセット
RegionZ	5	2Dポリゴンデータセット
RegionZ	105	3Dポリゴンデータセット
Text	7	テキストオブジェクト
CAD	149	複数のジオメトリオブジェクトを格納する複合データセット
Network	4	2Dネットワークデータセット
Network3D	205	3Dネットワークデータセット
Model	20	3Dモデルデータセット
Image	88	マルチバンド画像データセット
Grid	83	Gridデータセットは、デジタルサーフェスモデルに対応し、ピクセル値は地表の特徴量を表します
VoxelGrid	89	ボクセルラスタデータセット
Mosaic	206	モザイクデータセット

1.4 Geometry**タイプ**

2D/3Dポイント、ラインおよびポリゴンデータセットに格納されているオブジェクトは、SpatiaLiteのポイント、ライン、ポリゴンオブジェクトを使用し、オブジェクトの命名は"GAIA"プレフィックスで識別され、その他のオブジェクトタイプは"Geo"というプレフィックスで識別されます。各Geometryタイプは表2に示され、対応するストレージ構造はセクション4を参照してください。

表2 Geometryタイプの説明

Geometryタイプ	列挙値	説明
GAIAPoint	1	2Dポイントオブジェクト
GAIAPolygon	3	2Dポリゴンオブジェクト
GAIAMultiLineString	5	2Dライン（サブオブジェクト含む）
GAIAMultiPolygon	6	2Dライン（サブオブジェクト含む）
GAIAPointZ	1001	3Dポイントオブジェクト
GAIAMultiLineStringZ	1005	3Dライン（サブオブジェクト含む）
GAIAMultiPolygonZ	1006	3Dポリゴン（サブオブジェクト含む）
GeoPoint3D	101	3Dポイント
GeoLine3D	103	3Dライン（サブオブジェクト含む）
GeoRegion3D	105	3Dポリゴン（サブオブジェクト含む）
GeoText	7	テキスオブジェクト
GeoModel3D	1218	3Dモデルオブジェクト
GeoRect	12	矩形/斜め矩形
GeoRectRound	13	角丸矩形
GeoCircle	15	円
GeoEllipse	20	楕円
GeoPie	21	扇形
GeoArc	24	円アーク
GeoEllipticArc.	25	楕円アーク
GeoCardinal	27	Cardinal曲線
GeoCurve	28	ペンシルライン
GeoBSpline	29	Bスプライン曲線

2　システムテーブル

システムテーブルは、データソース内のデータセット情報を格納および管理するために使用されます。システムテーブルはSpatiaLiteで定義されたシステムテーブルとSuperMapによって定義されたシステムテーブルに区分されます。

2.1 SpatiaLiteシステムテーブル

UDBXはSpatiaLiteを使用してポイント、ライン、ポリゴンデータセットを管理するため、主にSpatiaLiteの座標系とベクタデータセット関連のシステムテーブルに関係しています。

2.1.1 座標系情報テーブル

座標系情報はspatial_ref_sysとspatial_ref_sys_auxに格納され、2つのテーブルはsridによって関連付けられます。

表3 spatial_ref_sysspatial_ref_sysのフィールド情報

フィールド名	タイプ	null値可否	説　　明
srid	INTEGER	N	主キー;座標系のユニークID
auth_nam	TEXT.	N	座標系の作成者/正式名称を定義します
auth_srid	INTEGER	N	座標系の内部ID
ref_sys_name	TEXT	N	座標系の名
proj4tex	TEXT	N	proj.4[1]テキスト形式で表される座標
srtext.	TEXT	N	wkt[2]で表される座標系

表4 spatial_ref_sys_auxのフィールド情報

フィールド名	タイプ	null値可否	説　　明
srid	INTEGER	N	主キー;外部キーは、spatial_ref_sys(srid)に関連付けられます。座標系のユニークID
is_geographic	INTEGER	Y	地理座標系かどうか
has_flipped_axes	INTEGER	Y	軸が反転するかどうか
spheroid	TEXT	Y	参照楕円体
prime_meridian	TEXT	Y	中央子午線
datum	TEXT	Y	測地系
projection	TEXT	Y	投影方法
unit.	TEXT	Y	座標系の単位
axis_1_name	TEXT.	Y	スピンドル名
axis_1_orientation	TEXT	Y	スピンドルが向いています
axis_2_name	TEXT	Y	副軸の名前
axis_2_orientation	TEXT	Y	副軸は向いています

2.1.2 ベクタデータセットシステムテーブル

SpatiaLiteのベクタデータセットシステムテーブル情報は、geometry_columns格納されます。

表5 geometry_columnsのフィールド情報を示します

フィールド名	タイプ	null値可否	説　　明
f_table_name	TEXT	N	データテーブルの名前
f_geometry_column	TEXT	N	データテーブルのgeometry列名。共用主キー(f_table_name,f_geometry_column).
geometry_type	INTEGER	N	geometry型を表2に示します
coord_dimension	TEXT	N	geometry座標のディメンション
srid	TEXT	N	座標系は、テーブルのspatial_ref_sysで識別されます sridフィールドの関連付け
spatial_index_enabled	INTEGER	N	空間インデックスが確立されているかどうか 値：0はインデックスなし、1はR^*ツリーインデックスを表します

2.2 SuperMap**システムテーブル**

SuperMapカスタムシステムテーブルは、データソース記述情報テーブル、ベクタデータセットシステムテーブル、ラスタデータセットシステムテーブルが含まれます。

2.2.1 データソースの説明情報テーブル

SmDataSourceInfoテーブルには、データソースの基本的な説明情報が格納されます(表6を参照)。

表6 SmDataSourceInfoテーブルのフィールド情報

フィールド名	タイプ	null値可否	説　　明
SmFlag	INTEGER	N	データソースID、主キー
SmVersion	INTEGER	Y	バージョン番号。現在のバージョン番号は10です
SmDsDescription	TEXT	Y	データソースの説明情報
SmProjectInfo	BLOB	Y	データソースの座標系情報については、4.7.1を参照
SmLastUpdateTime	DATE	N	データソースが最後に更新された時刻

SmDataFormat	INTEGER	N	データ保存フォーマット。現在の値は0で、UTF8エンコードによる表します

2.2.2 ベクトルデータセットシステムテーブル

ベクタデータセットシステムテーブルには、データセットレジストリテーブル、フィールド情報テーブル、値域情報テーブルが含まれます。

2.2.2.1 ベクタデータセットレジストテーブル

ベクタデータセットの登録情報は、SmRegisterテーブルに記録され、データセット名、対応するテーブル名、データセットの種類、親子データセットの関係などを含みます(表7を参照)。

表7 SmRegisterテーブルのフィールド情報

フィールド名	タイプ	null値可否	説　　明
SmDatasetID	INTEGER	N	主キー;データセットID
SmDatasetName	TEXT	Y	データセット名
SmTableName	TEXT	Y	データテーブル名
SmOption	INTEGER	Y	データセットのオプション情報、レコードがピラミッド付きかどうか、圧縮されているかどうか等、内部的に使用されます
SmEncType	INTEGER	Y	予約フィールド
SmParentDTID	INTEGER	N	親データセットIDは空にすることができます
SmDatasetType	INTEGER	Y	データセットの種類、列挙値は表1を参照
SmObjectCount	INTEGER	N	オブジェクトの数(データテーブルのレコード数)
SmLeft	REAL	Y	データセットの地理的範囲：
SmRight	REAL	Y	データセットの地理的範囲：右
SmTop.	REAL	Y	データセットの地理的範囲：下
SmBottom	REAL	Y	データセットの地理的範囲：上
SmIDColName	TEXT	Y	データテーブルオブジェクトID列名
SmGeoColName	TEXT	Y	データテーブル幾何オブジェクトの列名
SmMinZ	REAL.	Y	データセットの最小高さは、3Dデータセットに適用されます
SmMaxZ	REAL.	Y	データセットの最大高さは、3Dデータセットに適用されます
SmSRID	INTEGER	Y	座標系IDは、spatial_ref_sysテーブルのsridに関連します。フィールド値がnullの場合SmProjectInfoの値を取得します。

SmIndexType	INTEGER	Y	空間インデックスタイプ、値の範囲{0,2},0はインデックスなし、2はR木インデックスを表します。
SmToleranceFuzzy	REAL	Y	ノートスナップトレランス；トポロジ処理/編集時に使用されます。
SmToleranceDAngle	REAL	Y	角度トレランス。トポロジ処理/編集時に使用されます。
SmToleranceNodeSnap	REAL	Y	ロングダングルトレランス。トポロジ処理/編集時に使用されます。
SmToleranceSmallPolygon	REAL	Y	ポリゴン積トレランス。トポロジ処理/編集時に使用されます。
SmToleranceGrain	REAL	Y	
SmMaxGeometrySize	INTEGER	N	幾何オブジェクトバイナリストリーム 最大バイト数
SmOptimizeCount	INTEGER	N	予約フィールド
SmOptimizeRatio	REAL	Y	予約フィールド
SmDescription	TEXT	Y	データセット記述情報
SmExtInfo	TEXT	Y	データセットユーザーカスタム拡張情報。
SmCreateTime	DATETIME	Y.	データセットが作成された時刻。
SmLastUpdateTime	DATETIME	Y.	データセットが最後に更新された時刻。
SmProjectInfo	BLOB	Y.	データセットの座標系(4.7.1を参照)。値を指定すると、データセットの座標系が優先され、データソースの座標系は無視されます。

2.2.2.2 データセットフィールド情報テーブル

ベクタデータセットのフィールド情報はSmFieldInfoテーブルに格納され、表8を参照して、各データセットのフィールド(データセットのプライマリテーブルのフィールドに対応する)、各フィールドのエイリアス、SuperMapに対応するフィールドの種類などの情報が主に記録されます。

表8 SmFieldInfoテーブルのフィールド情報

フィールド名	タイプ	NULL値可否	説　明
SmID	INTEGER	N	主キー
SmDatasetID	INTEGER	Y	所有するデータセットのID SmRegisterのSmDatasetIDフィールドに対応します。

SmFieldName	TEXT	Y	フィールド名
SmFieldCaption	TEXT	Y	フィールド識別名
SmFieldType	INTEGER	Y	SuperMapに対応するフィールドのタイプ 表9SuperMapのフィールドタイプに示します
SmFieldFormat.	TEXT	Y	フィールドの値のフォーマット文字列
SmFieldSign	INTEGER	Y	フィールドIDを表10に示します
SmFieldDomain	TEXT	Y	破棄されたフィールド
SmFieldUpdatable	INTEGER	Y	フィールド値を変更可能かどうか
SmFieldbRequired	INTEGER	Y.	フィールドを入力するかどうか
SmFieldDefaultValue	TEXT.	Y	フィールドのデフォルト値
SmFieldSize	INTEGER	Y	フィールドの長さ

表9 SuperMapのフィールド種類

列挙値	タイプ	バイト数	値域	説　明
0	Unknown	/		無効値
1	Boolean	1	0^1	ブールタイプ
2	Byte	1	[0,255]	符号なしバイト
3	Int16	2	[-32768,32767]	16ビット整数型
4	Int32	4	[-2147483648,2147483647]	32ビット整数型
16	Int64	8	$[-2^{63},(22^{63}-1)]$.	64ビット整数型
6	Float	4	$[-3.4\times10]38,3.4\times10^{38}]^{38}$	単精度型
7	Double	8	$[-1.7\times10]308,1.7\times10^{308},1.7\times10^{308}]$	倍精度型
10	Text	外部指定	/	不定長の文字列
127	NText.	外部指定	/	ワイドバイト不定長の文字列
18	Char	外部指定	/	固定長文字列
8	Date	/	/	日付：年、月、日。時間なし
22	Time	/	/	時刻：時間、分、秒。日付なし
2	TimeStamp	/		タイムスタンプ、年、月、日、時、分、秒
9	Binary	外部指定	/	固定長バイナリ型
11	LongBinary	外部指定	/	不定長バイナリ型

表10 SuperMapのフィールドID

列挙値	説　　明
0	通常のフィールド
1	ネットワークデータセットのノードID(SmNodeIDフィールドをデフォルト)
2	ネットワークデータセットのエッジの起点フィールド(SmFNodeフィールドをデフォルト)
3	ネットワークデータセットのエッジの終点フィールド(SmTNodeフィールドをデフォルト)
4	ネットワークデータセットエッジのIDフィールド
11	オブジェクトIDフィールド
12	幾何オブジェクトフィールド
50	ユーザー定義フィールド開始値を識別します

2.2.2.3 値域情報テーブル

値域は、範囲値域(表11を参照)と列挙値域(表12を参照)に分割され、フィールドの値を制約します。

SmDomainsテーブルは、2種類の値域情報をまとめています。表13参照;SmDomainFieldはフィールド及び適用される値域規則を記録しています(表14を参照)。

表11 SmRangeDomainsテーブルのフィールド情報

フィールド名	タイプ	null値可否	説　　明
DomainID	INT	N	主キー
FieldType	INT.	N	フィールドタイプ
DomainRangeInfos	BLOB	Y	範囲値域規則オブジェクト(4.7.2を参照)

表12 SmCodeDomainsテーブルのフィールド情報

フィールド名	タイプ	null値可否	説　　明
DomainID	INT	N	主キー
FieldType	INT	N	フィールドタイプ
DomainCodeInfos	BLOB	Y	値域規則オブジェクトを列挙します(4.7.2を参照)

表13 SmDomainsテーブルのフィールド情報

フィールド名	タイプ	null値可否	説　　明
DomainID	INT	N	主キー;値域規則ID
DomainName	TEXT	N	値域名
DomainDescription	TEXT	N	説明情報
DomainType	INT	N	値域タイプ,更新： 1は範囲値域を表し、範囲内では有効 3は範囲値域を表し、範囲内ではないと有効 2は有効な列挙値域を表し、列挙値は有効 4は無効な列挙値域を表し、列挙値は無効

表14 SmDomainFieldテーブルのフィールド情報

フィールド名	タイプ	null値可否	説　　明
DatasetID	INT	N	データセットID; SmRegisterテーブルのSmDatasetIDフィールドに関連
FieldName	TEXT	N	値域規則が適用されるデータセットのフィールド名 主キー(DatasetID、FieldName)を結合し、SmFieldInfoテーブルの . (SmDatasetID、SmFieldName)関連付け、表8を参照

2.2.3ラスタデータセットシステムテーブル

ラスタデータセットは2つのシステムテーブルで管理する：SmImgRegisterがラスタデータセット情報を格納し、SmBandRegisterがラスタデータセットのバンド/レイヤー情報と対応するピラミッドデータセット情報を格納します。

2.2.3.1ラスタデータセットレジストリ

SmImgRegisterのフィールド情報を表15に示します。

表15 SmImgRegisterテーブルのフィールド情報

フィールド名	タイプ	null値可否	説　　明
SmDatasetID	INTEGER	N	主キー; データセットID
SmDatasetName	TEXT	N	データセット名
SmTableName	TEXT	N	データセットテーブル名
SmDatasetType	INTEGER	N	データセットタイプ、列挙値は表1に示されています
SmWidth	INTEGER	Y	データセットの幅、ピクセル単位
SmHeight.	INTEGER	Y	データセットの高さ、ピクセル単位
SmeBlockSize	INTEGER	Y	保存ブロックの大きさはピクセル単位として、長さ一致。設定64|128|256|1024
SmColorSpac。	INTEGER	Y	破棄されたフィールド
SmGeoLeft.	REAL	Y	地理的範囲：左
SmGeoTop.	REAL	Y	地理的範囲：上
SmGeoRight	REAL	Y	地理的範囲：右
SmGeoBottom	REAL	Y	地理的範囲：下
SmCreateTime	DATE	N	データセットが作成された時刻
SmCreato	TEXT	N	作成者
SmDescription	TEXT	Y	データセット記述情報
SmClipRegion	BLOB	Y	クリッピングポリゴン;表示時にポリゴンの内側の領域のみが表示されます。
SmExtInfo	TEXT	Y	データセット拡張記述情報
SmStatisticsInfo	TEXT	Y	統計情報
SmProjectInfo	BLOB	Y	座標系情報、4.7.1を参照

2.2.3.2バンド情報レジストリ

表16 SmBandRegisterテーブルのフィールド情報

フィールド名	タイプ	null値可否	説　　明
SmBandID	INTEGER	N	主キー;バンドID
SmDatasetID	INTEGER	N	所属データセットIDとSmImgRegisterの SmDatasetIDを関連付け
SmBandIndex.	INTEGER	N	バンドの順序番号
SmBandName	TEXT	N	バンド名
SmBandAvail	INTEGER	N	使用可能かどうか、値1：可用、0：不可用
SmOption	INTEGER	Y	データセットオプション情報、内部使用、レコードにピラミッドなどの情報が含まれるかどうか記録
SmScalar	INTEGER	Y	予約フィールド
SmEncType	INTEGER	N	Blockの圧縮符号化方法を表17に示す
SmPixelFormat.	INTEGER	N	ピクセル形式を表18に示します
SmMaxBlockSize	INTEGER	Y	ストレージブロックの最大サイズ、単位、バイト
SmMinZ	REAL	Y	ピクセル値の最小値
SmMaxZ	REAL.	Y	ピクセル値の最大値
SmAltitude	REAL	Y	レイヤーデータは高さを表します、外部設置
SmPyramid	TEXT	Y	親データセット名、空に設定することができないなら、ピラミッドデータセットに
SmPyramidLevel	INTEGER	N	ピラミッドレイヤーID;0より大きい場合は対応するピラミッドレイヤー
SmCreator	TEXT.	N	作成者
SmCreateTime	DATE	N	作成時刻
SmNovalue	REAL	Y	Null値
SmPalette	BLOB	Y	カラーパレット(カラーコントロールテーブル)は、視覚化時に使用されます。Color配列として格納されます(4.1.10を参照)

表17 Block圧縮エンコード方法

列挙値	説　　明	精度の損失
0	圧縮エンコーディングは使用されません	ロスレス
8	DCT、離散コサイン圧縮	破損している
9	SGL、実行符号化圧縮	ロスレス
11	LZW圧縮	ロスレス
12	PNG圧縮	ロスレス

表18 ピクセル形式

列挙値	説　　明
1	MONO、単一値
4	4ビットの数値
8	8ビット符号なし
80	8ビットは符号付き
16	16ビットは符号付き
160	16ビット符号なし
24	24ビットの真の色
32	32ビットは、真の色を強化
320	32ビットには符号付きのステレオタイプがあり
321	32ビット符号なしの整形
64	64ビットには、符号付きロングプロファイルがあり
3200	32ビット浮動小数点型
6400	64ビット倍精度浮動小数点型

3　データテーブル

データテーブルは、実際のデータを格納するために使用され、各データセットは1つ以上のテーブルに対応し、システムテーブルの管理範囲によって、ベクタデータセットとラスタデータセットの2つのカテゴリに分類されます。

3.1 **ベクタデータセット**

ベクタデータセットには、属性データセット、2D/3Dポイント/ライン/ポリゴンデータセット、テキストデータセット、複合データセット、2D/3Dネットワークデータセット、3Dモデルデータセット、およびビデオモザイクデータセットが含まれます。

ベクタデータセットの空間インデックスは、SQLiteの空間インデックスメカニズムを使用します。関連説明は公式ドキュメントが参照できます。

3.1.1 属性データセット

属性データデータセットには空間データが格納されません、データテーブルに対応するシステムのフィールドは表19に示されています。

表19 属性データ・セット・システム・フィールド

フィールド名	タイプ	null値可否	説　　明
SmID	INTEGER	N	主キー;オブジェクトのユニークID
SmUserID	INTEGER	Y	ユーザーカスタムID値

3.1.2 2 D/3D/ポイントデータセット

2Dポイントデータセットと3Dポイントデータセットシステムフィールドは同じです(表20を参照)。ここで、SmGeometryフィールドストレージポイントオブジェクトのタイプは、geometry_columnsのgeometry_typeフィールドによって決定されます(表5を参照)。

表20 ポイントデータセットシステムフィールド

フィールド名	タイプ	null値可否	説　　明
SmID	INTEGER	N	主キー;オブジェクトのユニックID
SmUserID	INTEGER	Y	ユーザーカスタムID値
SmGeometry	POINT	N	GAIAPointまたはGAIAPointZオブジェクトを保存し、保存構造は4.2.1および4.2.2を参照してください

3.1.3 2D/3D/ラインデータセット

2Dラインデータセットと3Dラインデータセットのシステムフィールドは同じです(表21を参照)。ここで、SmGeometryフィールドストレージラインオブジェクトの具体的なタイプは、geometry_columnsのgeometry_typeフィールドによって決定されます(表5を参照)。

表21 行データ・セット・システム・フィールド

フィールド名	タイプ	null値可否	説　　明
SmID	INTEGER	N	主キー;オブジェクトのユニックID
SmUserID	INTEGER	Y	ユーザーカスタムID値
SmLength	REAL	N	ラインオブジェクトの長さ（単位はメートル）
SmTopoError	INTEGER	N	トポロジトレランス、ラインデータセットトポロジ時使用
SmGeometry	MULTILINESTRING	N	GAIAMultiLineStringまたはGAIAMultiLineStringZオブジェクトの保存、保存構造は4.2.3および4.2.4を参照してください

3.1.4 2D/3D/ポリゴンデータセット

2Dポリゴンデータセットと3Dポリゴンデータセットシステムフィールドは同じです(表22を参照)。ここで、SmGeometryフィールドストレージフェイスオブジェクトの具体的なタイプは、geometry_columnsのgeometry_typeフィールドによって決定されます(表5を参照)。

表22 ラインデータ・セット・システム・フィールド

フィールド名	タイプ	null値可否	説　　明
SmID	INTEGER	N	主キー;オブジェクトのユニックID
SmUserID	INTEGER	Y	ユーザーカスタムID値
SmArea	REAL	N	ポリゴンオブジェクトのポリゴン積,単位は平方メートル
SmPerimeter	REAL	N	ポリゴンオブジェクトの周長;単位はメートル
SmGeometry	MULTIPOLYGON	N	GAIAMultiPolygonまたGAIAMultiPolygonZオブジェクトを保存し、保存構造については、セクション4.2.6および4.2.7を参照してください

3.1.5 テキストデータセット

テキストデータセットシステムフィールドは表23に示され、1つのテキストデータセットは1つのデータテーブルに対応します。

表23 テキストデータセットシステムフィールド

フィールド名	タイプ	null値可否	説　　明
SmID	INTEGER	N	主キー;2Dテキストオブジェクトのユニック ID
SmUserID	INTEGER	Y	ユーザーカスタムID値
SmGeometry	BLOB	Y	GeoTextのバイナリストリームデータについては、セクション4.3.8を参照してください
SmIndexKe	POLYGON	Y	オブジェクトの範囲、GAIAPolygonオブジェクトとして保存。4.2.5セクションを参照、空間インデックスのメンテナンスに使用されます

3.1.6 複合データセット

複合データセットにはさまざまな幾何オブジェクトタイプを保存できます。セクション4.3を参照。1つの複合データセットは、一つのデータテーブルに対応します、表24に示す。

表24　複合データセットシステムフィールドを示します

フィールド名	タイプ	null値可否	説　　明
SmID	INTEGER	N	主キー;オブジェクトのユニック ID
SmUserID	INTEGER	Y	ユーザーカスタムID値
SmGeoType	INTEGER	N	Geometryタイプを表25に示します
SmGeometry	BLOB	Y	SuperMapシンプルオブジェクトのバイナリストリームについては、4.3節を参照してください
SmIndexKey	POLYGON	Y	SuperMapシンプルなオブジェクトの範囲については、4.2.5節を参照してください、空間インデックスのメンテナンスに使用されます

表25複合データセットに保存されているオブジェクトの種類

Geometry型	列挙値	説　　明
GeoPoint	1	2Dポイント
GeoLine	3	2Dライン(サブオブジェクトを持つ可能)
GeoRegion	5	2Dポリゴン(サブオブジェクトを持つ可能)
GeoText	7	テキスト(4.4節を参照)
GeoRect	12	矩形/斜め矩形
GeoRectRound	13	丸角矩形
GeoCircle	15	円

GeoEllipse	20	楕円
GeoPie	21	扇ポリゴン
GeoArc	24	円アーク
GeoEllipticArc.	25	楕円アーク
GeoCardinal	27	Cardinal曲線
GeoCurve	28	ペンシルライン
GeoBSpline	29	Bタイプ曲線
GeoPoint3D	101	3Dポイント
GeoLine3D	103	3Dポイント(サブオブジェクトを持つ可能)
GeoRegion3D	105	3Dポリゴン(サブオブジェクトを持つ可能)

3.1.7 2D/3Dネットワークデータセット

2Dネットワークデータセットと3Dネットワークデータセットの保存方式は同じです。プライマリテーブルとサブテーブルで構成され、プライマリテーブルにはネットワークデータセットのエッジとノード接続情報が保存され、サブテーブルにはネットワークデータセットのノードが保存されます(表26および表27を参照)。

3Dネットワークデータセットなら、プライマリテーブルのSmGeometryフィールドには3Dラインオブジェクトが保存され、サブテーブルは3Dポイントオブジェクトを保存します。

表26 ネットワークデータセットのプライマリテーブルのシステムフィールド

フィールド名	タイプ	null値可否	説　明
SmID	INTEGER	N	主キー;ラインオブジェクトのユニックID
SmUserID	INTEGER	Y	ユーザーカスタムID値
SmEdgeID	INTEGER	N	エッジID
SmFNode	INTEGER	Y	起点のポイントID
SmTNode	INTEGER	Y	終点結節ID
SmResistanceA	REAL	Y	正方向バリア
SmResistanceB	REAL	Y	負方向バリア
SmTopoError	INTEGER	N	トポロジトレランス、ラインデータセットトポロジ処理時使用
SmLength	REAL	N	ラインオブジェクトの長さ、単位はメートル
SmGeometry	MULTILINESTRING	N	GAIAMultiLineStringまたはGAIAMultiLineStringZオブジェクト。保存構造は4.2.3および4.2.4を参照してください

表27 ネットワークデータセットのサブテーブルのシステムフィールド

フィールド名	タイプ	null値可否	説　　明
SmID	INTEGER	N	主キー;ポイントオブジェクトのユニックID
SmUserID	INTEGER	Y	ユーザーカスタムID値
SmNodeID	INTEGER	Y	ノードID
SmGeometry	POINT	N	GAIAPointまたはGAIAPointZオブジェクト、保存構造は4.2.1および4.2.2を参照してください

3.1.8 3Dモデルデータセット

3Dモデルデータセットには、3DモデルオブジェクトGeoModel3D、オブジェクト論理構造、および構造4.3セクションが格保存れます。

1つの3Dモデルデータセットは、メインテーブルとサブテーブルの2つのデータテーブルに対応し、表28と表29を参照してください。プライマリテーブルにはモデルの構造情報が保存され、サブテーブルにはモデルに関連付けられた実体オブジェクトが保存されます。メインテーブルに保存されているモデル構造は、モデルオブジェクトが参照される実体オブジェクト名、実体オブジェクト名に基づく64ビットHashCodeエンコーディングを記録し、サブテーブルのSmHashCodeフィールドに保存されます。

表28 3Dモデルデータセットのメインテーブルのシステムフィールド

フィールド名	タイプ	null値可否	説　　明
SmID	INTEGER	N	主キー;モデルオブジェクトのユニックID
SmUserID	INTEGER	Y	ユーザーカスタムID値
SmGeometry	BLOB	Y	GeoModel3Dのバイナリストリームデータについては、4.5.1を参照してください
SmIndexKey	POLYGON	Y	モデルオブジェクトの範囲(4.2.5セクションを参照)空間インデックス保守時使用

表29　3Dモデルデータセットサブテーブルのシステムフィールド

フィールド名	タイプ	null値可否	説　　明
SmID	INTEGER	N	主キー;モデルオブジェクトのユニックID
SmHashCode	INTEGER	Y	ユニック制約
SmGeometry	BLOB	Y	実体オブジェクトのバイナリストリームデータについては、セクション4.5.3を参照してください

3.2 ラスタデータセット

ラスタデータセットには、Imageデータセット、Gridデータセット、ボクセルグリッド(VoxelGrid)データセット、モザイクデータセットなどの2Dまたは3D空間のフィールドデータに保存されます。

Imageデータセット、Gridデータセット、およびVoxelGridデータセットのデータ編成と保存方式は同じです。

(1)元データ

元のデータ保存は一つのデータテーブルに対応します，メインテーブルという。保存される基本単位はブロック(Block)で、ブロックのサイズはSmImgRegisterのSmeBlockSizeフィールドに記録され、表15を参照してください。各ブロックは、メインテーブルのレコードの行に対応する固定長および幅のマトリックスデータブロックを表します。表30を参照してください。

(2)ピラミッド

ピラミッドデータは、ピラミッドの各層に1つのサブデータセットを持つ複数の層を持つ場合があり、各サブデータセットはプライマリデータセットと同じタイプのラスタデータセットであり、その親子関係はシステムテーブルSmBandRegisterによって維持されます(表16を参照)。

表30 グリッドデータテーブルのフィールド情報

フィールド名	タイプ	null値可否	説　　明
SmRow	INTEGER	N	ブロックの行番号
SmColumn	INTEGER	N	ブロックの列番号
SmBandID	INTEGER	N	グリッドデータ層ID Gridデータセットの場合、既定値は0です Imageデータセットの場合はバンド番号、ボクセルラスタデータセットの場合は層ID 合成主キー：(SmRow,SmColumn,SmBandID).
SmSize	INTEGER	N	ブロックデーバイトストリーム大きさ
SmBand	LONGBLOB	Y	ブロックデータ(保存の内容は4.6セクションを参照)

モザイクデータセットは、元データ+元の画像ファイルを使用して、3.2.4セクションに示す大画像データを管理します。

3.2.1 Imageデータセット

Imageデータセットは、1つ以上のバンドを持つ画像データを保存するために使用され、ピクセル値はカラー値を表します。

単バンドImageデータセットには、黒と白のグレースケール値、またはRGB合成ピクセル値のカラー値を保存できます。

画像データは一般的に大きく、通常はDCT(DiscreteCosineTransform)を使用します。DCTは離散余弦コードであり、高い圧縮率と性能を有する画像圧縮に広く用いられている変換コード方法であるが、データ精度は低下になる。データの精度を維持する場合はPNG、LZW圧縮コード(4.6セクションを参照)を使用できます。

3.2.2 Gridデータセット

Gridデータセットには、DTM(DigitalTerrainModel，数字地表モデル)データが保存されます。ピクセル値は、標高値、土壌タイプ、土地利用タイプ、岩盤の深さなどの地理または地理現象を表します。Gridデータセットに基づいて、グリッドデータ統計や代数演算などの数学的分析とグラフィックス処理に基づいて計算を行うことができます。

Gridデータセットは、LZWおよびSGLとして圧縮コードが使用できます(4.6節を参照)。SGLは、SuperMapによってカスタマイズされた圧縮ストレージ形式であり、重複データだけでなく、重複しないデータに対して圧縮操作を行う改善したLZW(Lempel_Zir_Welch)コードであります。

3.2.3 VoxelGridデータセット

VoxelGridデータセットは、空中電磁信号フィールド、空気および水質汚染フィールド、地下地質特性フィールドなどの連続的で非均質な3D空間属性フィールドデータを保存します。

連続する3次元フィールドデータをデータの異なる層にリサンプリングまたは補間し、各層のデータをブロックで保存し、高さ特性値の識別は、SmBandRegisterテーブルのSmAltitudeフィールドに記録されます(表16を参照)。

3.2.4 モザイクデータセット

モザイクデータセットのマスターテーブルはSmImgRegterシステムテーブルに記録され、各モザイクデータセットには、輪郭、境界、およびクリッピングを表す3つのサブデータセットがフックされます。3つのサブデータセットはすべてポリゴンデータセットであり、テーブルの名前付け規則は(tablenameはモザイクデータセットのプライマリテーブル名です)。

アウトラインサブデータセット：tablename_F境界サブデータセット：tablename_Bサブデータセットの切り抜き：tablename_C。

アウトラインサブデータセットは、モザイクデータセットが画像ファイルを保存および整理するための基盤であり、各ポリゴンを持つポリゴンデータセットです。

単一画像の地理的範囲と同様に、画像の分布とカバレッジをプロファイルでグローバルに参照できます(表31を参照)。

表31 モザイクデータセットのアウトラインサブデータセットテーブル

フィールド名	タイプ	null値可否	説　　明
SmID	INTEGER	N	主キー;オブジェクトID
SmUserID	INTEGER	Y	ユーザーカスタムID値
SmArea	REAL.	N	ポリゴンのポリゴン積
SmPerimeter	REAL	N	ポリゴンの周長
SmGeometry	MULTIPOLYGON	N	幾何オブジェクト
SmFileName	TEX	Y	フックされたファイルの名前
SmMinPS	REAL	Y	画像ファイルの最小表示解像度
SmMaxPS	REAL	Y	画像ファイルの最大表示解像度
SmLowPS	REAL	Y	画像の元の解像度
SmHighPS	REAL	Y	画像ピラミッドの解像度
SmCategory	INTEGER	Y	元のファイルかどうか。値を取る 1　　は元のファイルであることを示します 2　　プレゼンテーションは概要ビューです
SmPath	TEXT	Y	画像ファイルパス
SmInfo	TEXT	Y	画像ファイルの統計情報
SmZOrder	INTEGER	Y	予約フィールド
SmFileHash	TEXT	Y	ファイルに対応するHash値
SmOverviewLevel	INTEGER	Y	ピラミッドレベル、元のファイルは0です

境界サブデータセットにはモザイクデータセットの表示範囲が保存され、テーブルのフィールドは2Dポリゴンデータセットと同じです。

クリッピングサブデータセットには、各画像の表示範囲が保存され、テーブルのフィールドには、2Dポリゴンデータセットのフィールドに加えて、FootprintID、INTEGERタイプ、およびアウトラインサブデータセットのSmIDフィールドに関連付けられます。

4　オブジェクト保存構造

このセクションでは、UDBX内のさまざまなオブジェクトのバイナリ保存構造について説明します。バイトはLittle-Endianです。つまり、下位バイトがメモリの下位アドレス側に配置されます。

4.1 基本タイプ

オブジェクトストアの構造を記述する基本的なタイプには、基本データタイプと一般的に使用されるオブジェクトタイプがあります。

4.1.1基本データタイプ

オブジェクト保存構造に関連する基本データタイプを表32に示します。

表32 基本データ及び説明

タイプ	フィール数	値範囲	説　　明
byte	1	[0,255]	1バイト
bool	1	0^1	ブールタイプ
int16	2	[-32768,32767]	ショートタイプ
uint16	2	[0,65535]	符号なしの短いプロファイル
int32	4	[-2147483648,2147483647]	フルタイプ
uint32	4	[0,4294967295]	符号なしの整形
int64	8	[-2^{63},(22^{63}-1)]	ロングタイプ
uint64	8	[0,(2^{64}-1)]	符号なしロングプロファイル
float	4	[-3.4×10]38,3.4×10^{38}]^{38}	単精度浮動小数点型
double	8	[-1.7×10]308,1.7×10^{308},1.7×10^{308}]	倍精度浮動小数点型
wchar	2	--	ワイド文字型

4.1.2 String

オブジェクト保存構造に関連する文字列データタイプは、Unicodeでコードされ たStringオブジェクトで記述され、文字セットはUTF8として定義され、保存構造は次のようになります。

```
String{
        int32     length;   //バイト数
        byte      str[length];        //データ内容

}
```

4.1.3 Point

x,yで表された2D座標点：

```
Point{ //point座標オブジェクトdouble        x;
        double    y;.
}
```

4.1.4 PointZ

x、y、zで表される3D座標点：

```
PointZ{//point座標オブジェクトdoublex;doubley;doublez;.
}
```

4.1.5 Rect

左上の点と右下の点で表される矩形：

```
        Rect{     //矩形
        PointpntLB;.        //左下角点
        PointpntRT;.        //右上角点
}
```

4.1.6 BoundingBox

バウンディングボックス、最大ベクトルと最小ベクトルで表します。

```
        BoundingBox       {
              PointZboxMax;          バウンディング
                                     ボックスの最大ベ
                                     クトル
              PointZboxMin;          バウンディング
                                     ボックスの最小ベ
                                     クトル
}
```

4.1.7 Ring

リング、ポイントによって首尾相互接続

```
        Ring{                                  ポイントで構成
                                               されるリング
```

```
        int32     numPoints;          点数
        Point[]   pnts[numPoints];    点座標
    }
```

4.1.8 RingZ

リング、３Dポイントによって首尾相互接続の環状を構成。

```
    RingZ{                                ポイントで構成
                                        されるリング
    int32           numPoints;          点数
    PointZ[]        pnts[numPoints];    点座標
    }
```

4.1.9 Vector3D

3Dベクトル、保存構造はPointZと同じで、4.1.4を参照してください。

4.1.10 Color

カラー、rgbaの4つのコンポーネントで構成されるuint32値。

```
    Color  {
    byte        a;                  //alpha
                                    値
    byte        b;                  //blue値
    byte        g;                  //green
                                    値
    byte        r;                  //red値
}
```

4.2 SpatiaLiteの単純なオブジェクト

SpatiaLiteの単純なオブジェクトタイプは、2D/3Dのポイント、ライン、ポリゴンタイプです。

GAIAInfoは、SpatiaLiteの各種単一オブジェクストアのヘッド情報で、次のように構成されます。

```
GAIAInfo{                           幾何オブジェクトの基本情報
    staticbyte                      バイト順：小端保存
    byteOrdering=1;.
    int32     srid;                 座標系ID
    Rect      mbr;                  オブジェクトの座標範囲
```

```
        staticbyte                      MBR終了ID
        gaiaMBR=0x7c;
}
```

4.2.1 GAIAPoint

SpatiaLiteの2Dポイントオブジェクト：

```
GAIAPoint{
        staticbyte                      バイナリストリームの開始タグ
        gaiaStart=0x00;.
        GAIAInfo        info;           幾何オブジェクトの基本情報
        staticint32     geoType=1;      GeometryタイプID
        Point   geopnt;                 ポイントオブジェクトの座標値
        staticbyte                      バイナリストリームの終了タグ
        gaiaEnd=0xFE;
}
```

4.2.2 GAIAPointZ

SpatiaLiteの3Dポイントオブジェクト：

```
GAIAPointZ{
        staticbyte                      バイナリストリームの開始タグ
        gaiaStart=0x00;.
        GAIAInfo        info;           ジオメトリオブジェクトの基本情報
        staticint32                     Geometry型ID
        geoType=1001;.
        PointZ  geoPntZ;                ポイントオブジェクトの座標値
        staticbyte                      バイナリストリームの終了タグ
        gaiaEnd=0xFE;
}
```

4.2.3 GAIAMultiLineString

SpatiaLiteの2Dマルチラインオブジェクト：

```
GAIAMultiLineString{
        staticbyte                      バイナリストリームの開始タグ
        gaiaStart=0x00;.
```

GAIAGeoInfo info; 幾何の基本情報
staticint32 geoType=5;. GeometryタイプID。
int32 numLineStrings;. サブオブジェクトの数
LineStringEntity[] lineStrings[numLineStrings];//LineString のジオメトリデータ staticbyte gaiaEnd=0xFE; //バイナリストリーム終了タグ
}
LineStringEntity{
staticbyte gaiaEntityMark=0x69;//サブオブジェクトID
staticint32 geoType=2; GeometryタイプID
int32 numPoints; 点数
Point[] pnts[numPoints]; 各点の座標値
}

4.2.4 GAIAMultiLineStringZ

SpatiaLiteの3Dマルチラインオブジェクト：.
GAIAMultiLineStringZ{
staticbyte gaiaStart=0x00; バイナリストリームの開始タグ
GAIAGeoInfo info; 幾何オブジェクトの基本情報
staticint32 geoType=1005; GeometryタイプID
int32 numLineStrings; サブオブジェクトの数
LineStringZEntity[]lineStrings[numLineStrings];//LineString のジオメトリデータ staticbyte gaiaEnd=0xFE; //バイナリストリーム終了タグ
}
LineStringZEntity{
staticbyte gaiaEntityMark=0x69;//サブオブジェクトID
staticint32 geoType=1002; //GeometryタイプID.
int32 numPoints; //点数
PointZ[] pnts[numPoints]; //各点座標値
}

4.2.5 GAIAPolygon

SpatiaLiteの2Dポリゴンオブジェクト：
GAIAPolygon{
staticbyte gaiaStart=0x00; バイナリストリームの開始タグ

```
        GAIAInfo        info;               幾何オブジェクトの基本情報
    PolygonDatadata;                        ポリゴンジオメトリデータ
        staticbyte      gaiaEnd=0xFE;       バイナリストリームの終了タグ
    }

    PolygonData{                            ポリゴンジオメトリデータ
        staticint32     geoType=3;          GeometryタイプID
        int32   numInteriors;               内円の数
        Ring    interiorRing[numInteriors]; 外円オブジェクト
        Ring[]  interiorRings[numInteriors];//内円オブジェクト
    }
```

4.2.6 GAIAMultiPolygon

SpatiaLiteの2Dマルチフェイスオブジェクト：

```
GAIAMultiPolygon{
        staticbyte      gaiaStart=0x00;         バイナリストリームの開始タグ
        GAIAGeoInfo     info;                   幾何オブジェクトの基本情報
        staticint32     geoType=6;              GeometryタイプID
        int32   numPolygon;                     サブオブジェクトの数
        ポリゴンエンティティ[]  ポ リ ゴ ン     サブオブジェクトデータ
 [numPolygon];
   }
   PolygonEntity{
        staticbyte      gaiaEntityMark=0x69;    サブオブジェクトID
        PolygonData     data;                   サブオブジェクトデータ
   }
```

4.2.7 GAIAMultiPolygonZ

SpatiaLiteの3Dポリゴンオブジェクト：

```
   GAIAMultiPolygonZ{
        staticbyte      gaiaStart=0x00;         バイナリストリームの開始タグ
        GAIAGeoInfo     info;                   幾何オブジェクトの基本情報
        staticint32     geoType=1006;           GeometryタイプID
        int32   numPolygon;                     サブオブジェクトの数
```

```
        PolygonEntity[]   polygons[numPolygon];        サブオブジェクトデータ
    }
    PolygonEntity{
        staticbyte        gaiaEntityMark=0x69;        サブオブジェクトID
        PolygonZData      data;                       サブオブジェクトデータ
    }
    PolygonZData{                                     PolygonZの幾何データ
    staticint32       geoType=1003;                   GeometryタイプID
    int32             numInteriors;                   内円の数
    RingZ             exteriorRing;                   外円オブジェクト
    RingZ[]  interiorRings[numInteriors];//内円オブジェクト
}
```

4.3 複合データセットに保存されているオブジェクト

複合データセットには、2D/3Dポイント/ライン/ポリゴン、パラメータ化などの空間オブジェクト(表25を参照)とテキストオブジェクトを保存できます(セクション4.4を参照)。他のデータセットとは異なり、複合データセットはオブジェクトスタイルをGeoHeaderとしてオブジェクトヘッドに保存できます。

```
    ジオヘッド      {.
        int32           geoType;      オブジェクトタイプを表25に示します
        int32           styleSize;    オブジェクトスタイルはバイト数を占用
        Style           Style;        スタイルコンテンツについては、4.3.1のセクションを参照してください
}
```

4.3.1 Style

Styleは、オブジェクトの寸法によってポイントシンボル、ラインシンボル、およびポリゴンフィルスタイルに分割されます(シンボルのIDはSuperMapのシンボルライブラリで使用する必要があります)。

4.3.1.1 StyleMarker

ポイントシンボル

```
    StyleMarker   {
    int32         length;                 バイトストリームの長さ
    int32         markerStyle;            シンボルライブラリ内のポイントシンボルのID
    int32         markerSize;             シンボルサイズ、精度0.1ミリメートル
```

int32	markerAngle;	シンボルの回転角度(0.1度単位)
color	markerColor;	シンボルの色
int32	markerwidth;	シンボルの幅
int32	markerHeight;	シンボルの高さ
byte	reservedLength;	予約バイトの長さ
byte[]	reservedData[reservedLength+4];	予約データ
byte	fillOpaqueRate;	フィル透明度
byte	fillGradientType;	グラデーションフィルのタイプ
int16	fillAngle;	フィル角度
int16	fillCenterOffsetX;	フィル中心位置水平オフセットの割合
int16	fillCenterOffsetY;	フィル中心位置垂直オフセットの割合
color	fillBackcolor;	フィル背景色

}

4.3.1.2 StyleLine

ラインシンボル

StyleLine{

int32lineStyle;//lineラインシンボルはマークライブラリにあるID番号int32lineWidth;//線の太さ、0.1mmColorlineColor;. //線の色bytereservedLength;//予約バイトの長さ

byte[] reservedData[reservedLength+4]; //予約データ

}

4.3.1.3 StyleFill

ポリゴンフィルスタイル

StyleFill {

int32	lineStyle;	マークライブラリ内のラインシンボルのID
int32	lineWidth;	線の太さ、精度0.1mm
color	lineColor;	線の色
int32	fillStyle;	フィルライブラリ内のシンボルのID
Color	fillForecolor;	フィル前景色
Color	fillBackcolor;	フィル背景色
byte	fillOpaquerate;	フィル透明度
byte	fillgGadientType;	グラデーションフィルタイプ
int16	fillAngle;	フィル角度、精度0.1°
int16	fillCenterOffsetX;	フィル中心位置水平オフセットの割合
int16	fillCenterOffsetY;	フィル中心位置垂直オフセットの割合
byte	reserved1Length;	予約バイトの長さ

byte[] reserved1Data[reserved1Length+4];データを予約します
byte reserved2Length; //予約バイトの長さ
byte[] reserved2Data[reserved2Length+4];データを予約します
}

4.3.2 GeoPoint

2Dポイントオブジェクト
GeoPoint{.
GeoHeader header;
Point pnt;. //点座標値
}

4.3.3 GeoLine

2Dラインオブジェクト
GeoLine{.
GeoHeader header;
uint32 numSub; サブオブジェクトの数
int32 subPointCount[numSub]; 各サブオブジェクト点数
Point[] pnts[allPntCount]; allPntCountは、すべてのサブオブジェクトの点座標数の和です
}

4.3.4 GeoRegion

2Dポリゴンオブジェクト
GeoRegion{
GeoHeader header;
uint32 numSub; サブオブジェクトの数
int32 subPointCount[numSub]; 各サブオブジェクト点数
Point[] pnts[allPntCount]; allPntCountは、すべてのサブオブジェクトの点座標数の和です
}

4.3.5 GeoPoint3D

3Dポイント
GeoPoint3D

GeoHeader header;
PointZ pnt; 点座標
}

4.3.6 GeoLine3D

3Dライン
GeoLine3D
GeoHeader haeder;
uint32 numSub; サブオブジェクトの数
int32 subPointCount[numSub]; //サブオブジェクトごとの点数
PointZ[] pnts[allPntCount]; //allPntCountは、すべてのサブオブジェクトの点座標数です
の和
}

4.3.7 GeoRegion3D

3Dポリゴン
Geoheader3D
GeoHeader header;
uint32 numSub; //サブオブジェクトの数
int32 subPointCount[numSub];//サブオブジェクトごとの点数
PointZ[] pnts[allPntCount]; allPntCountは、すべてのサブオブジェクトの点座
} 標数の和

4.3.8 GeoRect

矩形オブジェクト
GeoRect {.
GeoHeader header;
point pntCenter; 中心位置座標値
Double width; 幅
double height; 高さ
int32 angle; 回転角度に10を掛け、丸め値を四捨五入
staticint32 reserved=0; 予約
}

4.3.9 GeoRectRound

丸角矩形

```
GeoRectRound    {
GeoHeaderheader;
Point    pntCenter;                中心位置座標値
double   width;                    幅
double   height;                   高さ
int32    angle;                    回転角度に10を掛け、丸め値を四捨五入しま。
staticint32        reserved=0;     予約
double   radiusX;                  丸角半長軸
double   radiusY;                  丸角半短軸
}
```

4.3.10 GeoCircle

円

```
GeoCircle       {
GeoHeaderheader;
Point    pntCenter;                中心位置座標値
double   radius;                   半径
}
```

4.3.11 GeoEllipse

楕円

```
GeoEllipse
{
GeoHeader          header;
Point              pntCenter;      中心位置座標値
Double             semimajoraxis;  半長軸
double             semiminoraxis;  半短軸
int32              angle;          回転角度に10を掛け、丸め値を四捨五入
```

staticint32 reserved=0; 予約
}

4.3.12 GeoPie

扇形

GeoPie {

GeoHeader	header;	
Point	pntCenter;	中心位置座標値。
Double	semimajoraxis;	半長軸
double	semiminoraxis;	半短軸
int32	rotationangle;	回転角度に10を掛け、丸め値を四捨五入
int32	startangle;	開始角度に10を掛け、丸め値を四捨五入
int32	endangle;	終了角度に10を掛け、丸め値を四捨五入
staticint32	reserved=0;	予約

}

4.3.13 GeoArc

円アーク

GeoArc {

GeoHeader	header;	
point	pntStart;	開始点の座標値
Point	pntMiddle;	中間点の座標値
point	pntEnd;	終端点座標値

}

4.3.14 GeoEllipticArc

楕円アーク

GeoEllipticArc {

GeoHeaderheader;

Point	pntCenter;	中心位置座標値
Double	semimajoraxis;	半長軸
Double	semiminoraxis;	半短軸
int32	rotationangle;	回転角度に10を掛け、丸め値を四捨五

```
        int32       startangle;         開始角度に10を掛け、丸め値を四捨
                                        五入
        int32       endangle;           終了角度に10を掛け、丸め値を四捨五入
        staticint32 reserved=0;         予約
}
```

4.3.15 曲線

曲線には、Cardinalカーブ、フリーカーブ、およびBスプラインが含まれます。保存構造は同じです。

```
    CurveObject     {
    GeoHeaderheader;
        uint32      numPnts;            曲線参照点の数
        Point[]     pnts[numPnts];      曲線参照点の座標
}
```

4.4 テキストオブジェクト

テキストオブジェクトは、テキストデータセットまたは複合データセットに保存できます。テキスデータセットに保存されている場合は、

```
GeoHeaderのstyleSizeは0
GeoText{
    GeoHeaderheader;        //オブジェクト先頭については、4.3サブセクションint32
        subCountを参照してください;        //textサブオブジェクトの数。
        TextStyletextStyle;//textスタイル
        GeoSubText      サブテキスト[subCount];   //textサブオブジェクト
    }
    GeoSubText{
    point           pntAnchor;//アンカーポイント
    int32           subAngle;//回転角度、実際の回転角度は10の丸め値で乗算され
                    ます
    staticint32     reserved=0;予約
    string          subText;.       //テキスサブオブジェクトのテキスト内容
    }
    TextStyle{.
    color               color;    //テキストカラー
    TextStyleBit        textStyleBit;       //テキストビットごとスタイル
    color               bgColor; //テキスト背景色
    Double              fontWidth;          //テキストフォント幅
```

```
Double          fontHeight;     //テキストフォント高さ
Point           pntAnchor;      //テキストアンカーポイント
string          faceName;       //テキストフォント名

TextStyleBit{
Byte            fixedSize;          固定サイズ
Byte            weight;             ストローク幅
byte            styleFlag;          太字斜体下線など
byte            alignFlag;          テキストの配置
}
```

styleFlagは、影付き、輪郭、背景不透明、固定サイズ、取り消し線、下線、斜体、太字など、8ビットで特定のスタイルで識別されるかどうかを示します。

alignFlagは2つの部分に分かれています,高い4ビット予約,低ビット4ビットは、フォントの配置を表し、値の意味を取る：

0：左上;1：中央上;2：右上;

6：左下;7：中下;8：右下;

9：左中央;10：中央、11：右中央

4.5 3Dモデルオブジェクト

3Dモデルオブジェクト(GeoModel3D)は、ローカル座標系を持つモデルオブジェクト(ModelNode)とその配置場所、姿勢などの情報で構成され、その組織構造は図2に示されています。

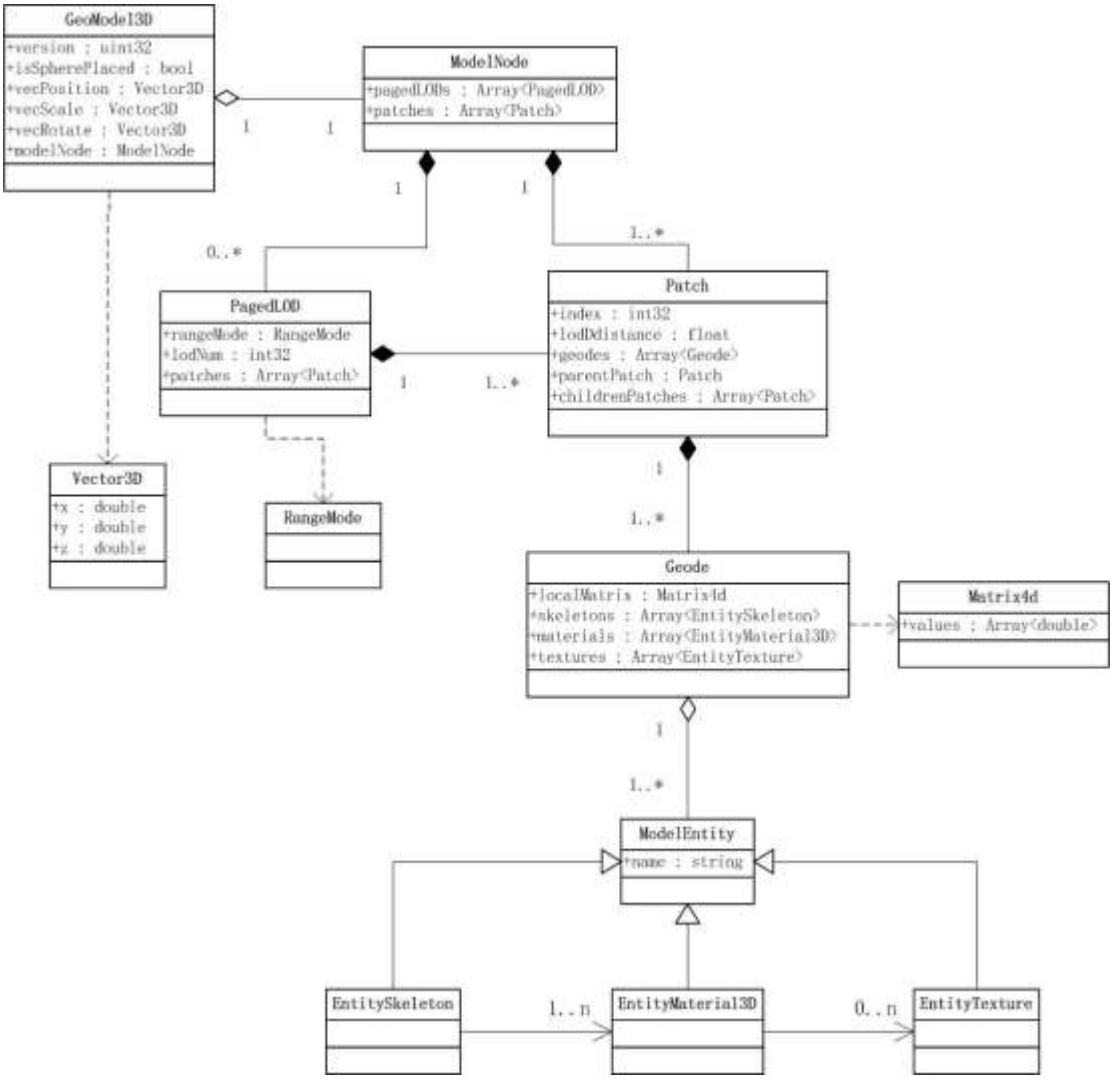

図2 GeoModel3Dオブジェクトの組織構造

ModelNodeは、精緻とLODレイヤー（PagedLODで表されるオプション）データで構成されます。ファインレイヤーとLODレイヤーの基本コンポーネントはPatchです。各パッチには複数のGeodeが含まれます。ジオードは、物理オブジェクトで構成されるデータパケットです（使用するModelEntity）、Geodeのマトリックスを介して、同じエンティティを異なる位置に配置して、モデルデータのインスタンス化ストレージを実現できます。

ModelEntityのサブクラスには、スケルトン（EntitySkeleton）、材料（EntityMaterial3D）、およびテクスチャが含まれます（EntityTexture）。

保存戦略の観点から、GeoModel3Dはメインテーブルに保存され、Geodeは実体オブジェクトの名前のみを保存します。実体オブジェクトはデータセットサブテーブルに個別に保存され、実体オブジェクト名に基づく64ビットHashCodeコードがオブジェクトIDとして使用されます。

4.5.1 GeoModel3D

GeoModel3D		
staticint32	type=1218;	オブジェクトタイプ
staticint32	hasStyle=0;	スタイルがあるかどうか
uint32	version;	オブジェクトバージョン番号
bool	isSpherePlaced;	球ポリゴンに配置するかどうか
Vector3D	vecPosition;	オブジェクト位置
Vector3D	vecScale;	オブジェクトスケール値
Vector3D	vecRotate;	オブジェクト回転値
BoundingBoxbbox;		バウンディングボックス
ModelNodemodelNode;		
}		

4.5.2 ModelNodeModelNode{

int32	numLODs;	LOD層数
PagedLOD pagedLODs[numLODs];		LOD層データ
int32	numPatches;	精緻モデルPatchの数
Patch;	patches[numPatches]	精緻モデルPatchデータ
}		
PagedLOD	{	
RangeMode	rangeMode;	int16として保存される範囲モードを切り替えます
int32	lodNum;	LOD層番号
int32	numPatches;	当該層のPatcheの数
Patch	patches[numPatches];	Patcheデータ
}		
EnumRangeMode { DISTANCE_FROM_EYE_POINT=0,		LODモード切り替えモード カメラまでの距離に応じて切り替え

PIXEL_SIZE_ON_SCREEN=1　　画ポリゴンのピクセルサイズに応じて切り替え
}

Patch{
float　　lodDdistance;　　距離を切り替え
int32　　index;　　現在データ層Patchのインデックス番号
int32　　parentIndex;　　親ノードのインデックス番号、-1は親ノードがないことを示します
int32　　numChildren;　　サブノードの数
int32　childrenIndexes[numChildren];//サブノードインデックス番号
int32　geodeCount;　　//Geodeの数
Geodegeodes[geodeCount]; //各Geodeデータ
}

Geode{
Matrix4dlocalMatrix;　　マトリックス情報
int32　numSkeletons;　　スケルトンの数
String skeletonNames[numSkeletons]。　　スケルトン名
int32　numMaterials;　　マテリアルの数
String materialNames[numMaterials].　　マテリアルの名前
int32　numTextures;　　テクスチャの数

String textureNames[numTextures] テクスチャ名

}

Matrix4d { //4*4行列、行の主順序

Doublevalues[16];

}

4.5.3 ModelEntity

4.5.3.1 EntitySkeleton

EntitySkeleton{

String name; スケルトン名

String materialName; 関連付けられたマテリアル名

BoundingBox bbox; バウンディングボックス

Matrix4d localMatrix; マトリックス

VertexDataPackage dataPack; 頂点データ

int32 numIndexpacks; インデックスパッケージの数

IndexPackage indexPacks[numIndexpacks]; インデックスパッケージの配列

}

VertexDataPackage{

VertexOptionsの列挙値をビット単位で演算した頂点プロパティ

int32 vertexOptions;

uint16 numDim; //頂点座標ディメンション

uint32numVertexes;//頂点の数uint16vertexStride;//頂点座標の配列内のオフセット

頂点座標データ。vertexOptionsにVO_VERTEX_DOUBLEプロパティがある場合はdouble、それ以外の場合はfloat

variant. vertexData[numVertexes*numDim];

uint32 numNormals; //法線ベクトル数

uint16 配列内のnormalStride //法線ベクトルオフセット

フロット normalData[numNormals*numDim]; //法線ベクトルデータ

uint32 numColors; //頂点カラー数

uint16 colorStride; //色は配列内でオフセット

uint32 colorData[colorCount]; //頂点カラー、4バイトストレージ R/G/B/A

int32 numTextures; //テクスチャチャネルの数

TextureCoordtextureCoords[numTextures];//テクスチャ座標データ

}

EnumVertexOptions{ //頂点データのプロパティ

VO_NORMALS=1、 //法線を含む

VO_TEXTURE_COORDS=2、//にはテクスチャ座標が含むVO_DIFFUSE_COLOURS=4、//には頂点カラーが含まれます

VO_SPECULAR_COLOURS=8、 //には頂点secondColorが含まれている必要があります

VO_BLEND_WEIGHTS=16, //重み値を使用して計算されます

VO_USE_SINGLE_COLOR= 32、//は1つの色で描画されます

VO_USE_POINT_SMOOTHING= 64、//ポイントアンチエイリアスを使用

VO_MATERIAL=128, //マテリアルを使用します

VO_TEXTURE_COLOR=256, //テクスチャカラーを使用します

VO_VERTEX_DOUBLE= 512、//頂点座標は高精度doubleです

VO_TEXTURE_COORD_Z_IS_MATRIX=1024の場合、//は頂点プロパティのZ値を表す行列です。

};

TextureCoord{ テクスチャ座標

uint16 dimension; テクスチャ座標ディメンション

uint32 numCoords; テクスチャ座標の数

uint16 stride; オフセット値

float coordData[numCoords*dimension]; //座標値

}

IndexPackage{

uint32 numIndexes; インデックスの数

IndexType. type; int32として保存されるインデックスデータタイプ

bool isUseIndex; インデックスを使用するかどうか

OperationType operationType; 頂点はint32として保存される方法で編成

インデックスデータ、typeがIT_32BITまたはIT_32BIT_2の場合、variantはuint32、さもなくば、uint16variantindexData[indexesCount

int32 numPass; //使用されるPassの数

String passNames[numPass];//で使用されるPassの名前の配列

}

EnumIndexType{

IT_16BIT=0, インデックス値はuint16で表され

IT_32BIT=1, インデックス値はuint32で表され

IT_16BIT_2=2, uint16で表されるインデックス値を持つプロパティインデックス

IT_32BIT_2=3, uint32で表されるインデックス値を持つプロパティインデックス

}

EnumOperationType{ 頂点の編成方法

OT_POINT_LIST=1, 単一のポイント

OT_LINE_LIST=2, 2点線

OT_LINE_STRIP=3, ライン文字列

OT_TRIANGLE_LIST=4, //三角形

OT_TRIANGLE_STRIP=5,//ストリップ三角形

OT_TRIANGLE_FAN= 6、//扇ポリゴン三角形で構成されます

OT_QUAD_STRIP= 8、//ストリップクワッド

OT_QUAD_LIST= 9、//クワッド文字列、エッジを共有しない

OT_POLYGON=10, //多角形

}

4.5.3.2 EntityMaterial3D

EntityMaterial3D{

Doubleversion;バージョン番号Stringname;材料名

String groupName;. //材料が属するグループ名

EffectType effectType;. //特殊効果材料タイプ(int32int32numTechniqueとして格納されます

int32 //Techniqueの数

Technique techniguea[numTechnique]; //Techniqueデータ

}

EnumEffectType{ //特殊効果材料列挙体

NONE=0, //特殊効果なし

WATER=1, 1、//ウォーターエフェクト

}

Technique{

String name; //Technique名

String schemeName; //Techniqueが属するschemeの名前

String lodIndex;//Techniqueで使用されるLOD層インデックス

String mShadowCasterMaterialName; シャドーキャストの材料名

String mShadowReceiverMaterialName; シャドー受け取る材料名

int32 numPass; passの数

Pass passes[numPass]; バインドされたすべてのpass

}

Pass{

String name; pass名

PolygonMode polygonMode; int32として保存される描画モード

CullingMode cullMode; トリミンモード(int32として格納)

Bool lightEnabled; ライトがオンかどうかを設定

uint32。 リザーブド; 使用されていません

bool リザーブド; 使用されていません

float pointSize; ポイントサイズ

float pointMinSize; ポイント最小サイズ

float pointMaxSize; ポイント最大サイズ

int16 リザーブド; 使用されていません

double リザーブド[3]; 使用されていません

SmoothHintModepntSmoothHintMode; ラインスムーズ方式はint32として保存

SmoothHintModelineSmoothHintMode; ポイントスムーズ方式はint32として保存

uint32 ambient; 環境光

uint32 diffuse; 散乱光

uint32 スペクター; 反射光

uint32 selfIllumination; 自己照明

uint32 materialColor; 材料カラー

floatshininess; 発光、放出された光点サイズに影響

uint32 tracking; 頂点カラートラッキング

bool receiveShadow; 影を受けるかどうか

bool colorWrite; 色を書き込むかどうか

float alphaReject; Alphaテスト参照値

CompareFunction アルファ Alphaテスト方式はint32として保存されます
RejectFunc;.

bool reserved; 使用されていません

bool transparentSorting; 透明なオブジェクトの深さの並べ替え

bool reserved; 使用されていません

bool depthCheck; 深度テストを実行するかどうか

bool depthWrite; レンダリング時にディープ書き込みを行うか
どうか

CompareFunctiondepthBufferFunc; int32として保存される深度テスト方法

float constantPolygonOffset; 多角形オフセット定数部分

float slopeScalePolygonOffset; 多角形オフセットの深度傾斜角度係数部分

float reserved; 使用されていません

bool blendAlpha; Alpha混合を行うかどうか

String vertexProgram; 頂点シェーダーの名前

String fragmentProgram; スライスメタシェーダーの名前

String geometryProgram; 幾何シェーダーの名前

String shadowCasterVertexProgram; シャドー投射頂点シェーダーの名前

String shadowReceiverVertexProgram; シャドー受取頂点シェーダーの名前

String shadowReceiverFragmentProgram; シャドー受取メタシェーダーの名前

int32 numTextureUnitState; テクスチャユニットの数

TextureUnitStatetextureUnitState[numTextureUnitState];//関連するテクスチャユニット

int32textureZType[numTextureUnitState];//各テクスチャZチャネル

}

EnumPolygonMode{ //レンダリングエンジンで使用されるポリゴン表示モード

PM_POINTS=1、//ポイントのみを表示します

PM_WIREFRAME= 2、//ワイヤフレームのみを表示します

PM_SOLID=3 //表示実体

}

EnumCullingMode{ //レンダリングエンジンで使用されるクリッピングモード

CULL_NONE=1, クリップは行わない

CULL_CLOCKWISE=2, 時計回りにクリップされます
CULL_ANTICLOCKWISE=3 反時計回りにクリップされます
}
EnumSmoothHintMode{ 画像描画のアンチエイリアスモード
SHM_NONE=0, アンチエイリアシングは使用されません
SHM_DONT_CARE=1, ポイント/ラインのスムーズ効果を達成するには、OpenGLが決定します
SHM_FASTEST=2, 最も速く走る
SHM_NICEST=3 表示が最適
}
EnumCompareFunction{ 各種試験の比較方法
CMPF_ALWAYS_FAIL=0, テストに合格しない
CMPF_ALWAYS_PASS=1, 常にテストに合格
CMPF_LESS=2, 参照値<バッファが値をマークした場合にのみ合格
CMPF_LESS_EQUAL=3, 参照値<=バッファが値をマークした場合にのみ通過
CMPF_EQUAL=4, 参照値=バッファーが値をマークした場合にのみ通過
CMPF_NOT_EQUAL=5, 参照値!=バッファが値をマークした場合にのみ通過
CMPF_GREATER_EQUAL=6, 参照値>=バッファが値をマークした場合にのみ合格
CMPF_GREATER=7 参照値>バッファが値をマークした場合にのみ合格
}
TextureUnitState{
String name; テクスチャユニットの状態名
String textureNameAlias; テクスチャエイリアス
String textureName; テクスチャユニットで使用されるテクスチャ名
String cubicTextureName; キューブテクスチャ名
uint32 reserved; 使用されていません
TextureAddressingModemodeU; テクスチャ座標アドレッシングモードU方向、int32として保存

TextureAddressingModemodeV;　　テクスチャ座標アドレッシングモードV方向、int32として保存

TextureAddressingModemodeW;　　テクスチャ座標アドレッシングモードW方向、int32として保存

FilterOptions　minFilter;　　縮小時のフィルタリングタイプ、バイトint32として保存、

FilterOptions　maxFilter;　　拡大時のフィルタリングタイプ、バイトint32として保存

FilterOptions　mipFilter;　　Mipmap時のフィルタリングタイプ、バイトint32として保存

double　UScale;　　テクスチャUのスケーリング

double　VScale;　　テクスチャVのスケーリング

bool　EnvironmentMapEnabled;//環境リマッピング·を有効にするかどうか

int32　reserved; //未使用

Matrix4d texModMatrix;//テクスチャマトリックス

}

EnumTextureAddressingMode{　　//テクスチャアドレッシングモード

TAM_WRAP//繰り返しテクスチャTAM_MIRROR//対称フリップ

TAM_CLAMP//周辺ピクセルは、1より大きいのテクスチャ座標をフィル

TAM_BORDER//[0,1]の範囲外のテクスチャ座標は、ユーザーが指定した周辺カラーを使用

}

EnumFilterOptions{　　//テクスチャまたはミップマップのフィルタリングモード

FO_NONE=0,　　フィルタリングなし

FO_POINT=1,　　最近いサンプリ

FO_LINEAR=2,　　線形サンプリ

FO_TRILINEAR=3, 三線性サンプリ

FO_ANISOTROPIC=4 線形サンプリと同様に、テクスチャ角度、異方性、お

} よび未使用は考慮されます

4.5.3.3 EntityTexture

EntityTexture{

Stringname; テクスチャ名

boolmipmap; mipmapを持っているかどうか

int32level; mipmapレベル

TextureDatatextureData; テクスチャデータ

}

TextureData{

staticuint32compressType=14; //テクスチャ圧縮タイプ,uint32として保存

uint32width;//テクスチャ幅uint32height;//テクスチャの高さ

PixelFormat format; //テクスチャピクセル形式int32として保存

uint32 size; //データストリームバイト長さint32 zipSize;. //zip圧縮後のサイズuchar data[zipSize]

//zip圧縮後のデータ

}

EnumPixelFormat{ //テクスチャピクセル形式

PF_BYTE_RGB=11、 //3バイトピクセル、各色が1バイトを占める

PF_BYTE_BGR= 10、//3バイトのピクセルで、各色は1バイトを占める

PF_BYTE_BGRA= 12、//4バイトのピクセル、各色とalphaが1バイトを占める

PF_BYTE_RGBA= 13、//4バイトピクセル、各色とalphaは1バイトを占める

}

4.6 グリッドブロックストレージ

グリッドブロック(Block)保存は、圧縮コード方法に関連しています(表17を参照)。

Blockバイナリストリームサイズはデータテーブルの SmSizeフィールドに書き込み、BlockデータがSmBand に書き込まれます。表30を参照してください。

1)非圧縮コード

Blockは、実際の長さ、幅、ピクセル単位で値を保存し、行順。ピクセル値のタイプと保存方法を表18に示します。

2)DCT圧縮

jpeglibオープンソースライブラリ(バージョン番号6b)を使用して、元のBlockデータストリームを圧縮します。

SGL圧縮

ブロックにはX方向に16ピクセル、Y方向に4ピクセルでTileを分割し、各Tileの内部でコードを実行します。

```
SGLBlock        {
        int16       numTiles;           Tileの数
        staticbyte  sizeX=16;           X方向のTileピクセル数
        staticbyte  sizeY=4;            X方向のTileピクセル数
        SGLTile[]   tileData[numTiles]; 各Tile圧縮後のデータ
}
SGLTile{
int16   dataLength;     //データストリームの長さ、バイト単位
byte    bitCountDiff;   //コード値では、Diff値占用桁数
byte    byteCountMin;   //Tileの最小値占用バイト数
        byte[]  minValueByByte[byteCountMin];   //Tileの最小値
        ValuePack[]     valuePacks[];//ランレングスコード値
}
ValuePack{
byte    tagCount;       //値の数。値の最大ビットが1の場合、値なし、低4ビットは値なし、values値なし；値の3桁目が1の場合、valuesはユニークな値であり、数値は下2桁に丸められます
        byte[]  values[bitCountDiff];       //元の値と最小値の差は、占有されているバイト数で書き込まれます
}
```

LZW圧縮は現在、zlibオープンソースライブラリ（バージョン番号1.2.2）を使用して、元のBlockデータストリームを圧縮します。

PNG圧縮は、libpngオープンソースライブラリ（バージョン番号1.2.6）を使用して、元のBlockデータストリームを圧縮します

4.7 **その他のオブジェクト**

4.7.1 座標系オブジェクト

座標系オブジェクトの保存構造は次のとおりです

```
ProjectInfo{
int32           prjCoordSysType;    投影座標系タイプ
int32           geoCoordSysType;    地理座標系のタイプ
int32           projectionType;     投影方法のタイプ
int32           datumType;          Datumタイプ
```

```
int32          spheroidType;          楕円体タイプ
    int32      primeMeridianType;     中央子午線タイプ
    int32      reserved;              予約
    int32      unit;                  座標系単位
    double     falseEasting;          水平オフセット
    double     falseNorthing;         垂直オフセット
    double     centralMeridian;       中央経線
    double     centralParallel;       原点緯線
    double     standardParallel1;     標準緯線1
    double     standardParallel2;     標準緯線2
    double     scaleFactor;           縮尺係数
    double     azimuth;               方位角
    double     firstPointLongitude;   第一点経線
double  secondPointLongitude;//第二点経線
    double     axis;                  楕円体半長軸
    double     flatten;               楕円体の扁平率
    double     primeMeridian;         中央子午線値
    double[]   reserved[2];           予約
    String     prjCoordSysName;       投影座標系名
    String     geoCoordSysName;       地理座標系名
    String     spheroidName;          楕円体名
    String     datumName;             datum名
    uint32     epsgCode;              EPSG番号
    double     rectifiedAngle;        修正角
}
```

関連する説明はOGDC、セクション5を参照してください。ここで、prjCoordSysType、geoCoordSysType、projectionType、datumType、spheroidType、primeMeridianTypeの列挙値に対応する意味は、Projection/UGPjCon.hファイルを参照できます。単位値unitはBase/ogdcdefs.hを参照できます

4.7.2値域規則オブジェクト

値域規則オブジェクトは、範囲値域(DomainRangeInfos)と列挙値域(DomainCodeInfos)に分類されます。

```
DomainRangeInfos        {
int32   numRanges;      //範囲間隔の数
DomainRange[]   ranges[numRanges];      //各間隔オブジェクト
}
DomainRange     {
DomainRangeType         rangeType;      //間隔タイプ
```

区間値タイプは、フィールドタイプによって決まります。

フィールドタイプInt16、Int32、variantはint32で対応

フィールドタイプInt64は、variantがint64で対応;//フィールド型タイプFloat、Doublは、variantがdouble

variant leftValue //区分左値variant rightValue;. //区分右値

}

```
EnumDomainRangeType {
    CloseClose      左値閉区分受取、右値閉区分受取
    =1,
    OpenClose       下限値に含む
    =2,
    CloseOpen       上限値に含む
    =3,
    OpenOpen        左開、右開
    =4
}
DomainCodeInfos{
int32               numCodes;           列挙値の数
DomainCode[]        ranges[numCodes];   各列挙オブジ
}                                       ェクト
DomainCode  {
```

値タイプは、フィールドのタイプによって決まります：

フィールドタイプBoolean、Byte、Int32、variantをint32に対応

フィールドタイプInt16、variantをint16に対応

フィールドタイプInt64、variantをint64に対応

フィールドタイプFloat、Double、variantをdoubleに対応

フィールドタイプText、NText、variantをStringに対応

```
    Variant  codeValue;        //列挙値
    String   codeDescription;  //列挙値説明
}
```