Well-known text representation of geometry

From Wikipedia, the free encyclopedia

"Well-known text" redirects here. For representation of coordinate reference systems, see Well-known text representation of coordinate reference systems.

Well-known text (**WKT**) is a text markup language for representing vector geometry objects. A binary equivalent, known as **well-known binary** (**WKB**), is used to transfer and store the same information in a more compact form convenient for computer processing but that is not human-readable. The formats were originally defined by the Open Geospatial Consortium (OGC) and described in their Simple Feature Access.[1] The current standard definition is in the ISO/IEC 13249-3:2016 standard.[2]

Geometric objects [edit]

WKT can represent the following distinct geometric objects:

- Point, MultiPoint
- LineString, MultiLineString
- Polygon, MultiPolygon, Triangle
- PolyhedralSurface
- TIN (Triangulated irregular network)
- GeometryCollection

Coordinates for geometries may be 2D (*x*, *y*), 3D (*x*, *y*, *z*), 4D (*x*, *y*, *z*, *m*) with an *m* value that is part of a linear referencing system or 2D with an *m* value (*x*, *y*, *m*). Three-dimensional geometries are designated by a "Z" after the geometry type and geometries with a linear referencing system have an "M" after the geometry type. Empty geometries that contain no coordinates can be specified by using the symbol `EMPTY` after the type name.

WKT geometries are used throughout OGC specifications and are present in applications that implement these specifications. For example, PostGIS contains functions that can convert geometries to and from a WKT representation, making them human readable.

The OGC standard definition requires a polygon to be topologically closed. It also states that if the exterior linear ring of a polygon is defined in a counterclockwise direction it will be seen from the "top". Any interior linear rings should be defined in opposite fashion compared to the exterior ring, in this case, clockwise.[3]

Geometry primitives (2D)

Type	Examples
Point	`POINT (30 10)`
LineString	`LINESTRING (30 10, 10 30, 40 40)`
Polygon	`POLYGON ((30 10, 40 40, 20 40, 10 20, 30 10))` `POLYGON ((35 10, 45 45, 15 40, 10 20, 35 10), (20 30, 35 35, 30 20, 20 30))`

Multipart geometries (2D)

Type	Examples
MultiPoint	`MULTIPOINT ((10 40), (40 30), (20 20), (30 10))` `MULTIPOINT (10 40, 40 30, 20 20, 30 10)`
MultiLineString	`MULTILINESTRING ((10 10, 20 20, 10 40), (40 40, 30 30, 40 20, 30 10))`
MultiPolygon	`MULTIPOLYGON (((30 20, 45 40, 10 40, 30 20)), ((15 5, 40 10, 10 20, 5 10, 15 5)))` `MULTIPOLYGON (((40 40, 20 45, 45 30, 40 40)), ((20 35, 10 30, 10 10, 30 5, 45 20, 20 35), (30 20, 20 15, 20 25, 30 20)))`
GeometryCollection	`GEOMETRYCOLLECTION (POINT (40 10), LINESTRING (10 10, 20 20, 10 40), POLYGON ((40 40, 20 45, 45 30, 40 40)))`

The following are some other examples of geometric WKT strings. (Note: Each item below is an individual geometry.)

```
GEOMETRYCOLLECTION(POINT(4 6),LINESTRING(4 6,7 10))
POINT ZM (1 1 5 60)
POINT M (1 1 80)
POINT EMPTY
MULTIPOLYGON EMPTY
TRIANGLE((0 0 0,0 1 0,1 1 0,0 0 0))
TIN (((0 0 0, 0 0 1, 0 1 0, 0 0 0)), ((0 0 0, 0 1 0, 1 1 0, 0 0 0)))
POLYHEDRALSURFACE Z ( PATCHES
    ((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0 0)),
    ((0 0 0, 0 1 0, 0 1 1, 0 0 1, 0 0 0)),
    ((0 0 0, 1 0 0, 1 0 1, 0 0 1, 0 0 0)),
    ((1 1 1, 1 0 1, 0 0 1, 0 1 1, 1 1 1)),
    ((1 1 1, 1 0 1, 1 0 0, 1 1 0, 1 1 1)),
    ((1 1 1, 1 1 0, 0 1 0, 0 1 1, 1 1 1))
  )
```

Well-known binary [edit]

Well-known binary (WKB) representations are typically shown in hexadecimal strings.

The first byte indicates the byte order for the data:

- `00` : big endian
- `01` : little endian

The next 4 bytes are a 32-bit unsigned integer for the geometry type, as described below:

Geometry types, and WKB integer codes

Type	2D	Z	M	ZM
Geometry	0000	1000	2000	3000
Point	0001	1001	2001	3001
LineString	0002	1002	2002	3002
Polygon	0003	1003	2003	3003
MultiPoint	0004	1004	2004	3004
MultiLineString	0005	1005	2005	3005
MultiPolygon	0006	1006	2006	3006
GeometryCollection	0007	1007	2007	3007
CircularString	0008	1008	2008	3008
CompoundCurve	0009	1009	2009	3009
CurvePolygon	0010	1010	2010	3010
MultiCurve	0011	1011	2011	3011
MultiSurface	0012	1012	2012	3012
Curve	0013	1013	2013	3013
Surface	0014	1014	2014	3014
PolyhedralSurface	0015	1015	2015	3015
TIN	0016	1016	2016	3016
Triangle	0017	1017	2017	3017
Circle	0018	1018	2018	3018
GeodesicString	0019	1019	2019	3019
EllipticalCurve	0020	1020	2020	3020
NurbsCurve	0021	1021	2021	3021
Clothoid	0022	1022	2022	3022
SpiralCurve	0023	1023	2023	3023
CompoundSurface	0024	1024	2024	3024
BrepSolid		1025		
AffinePlacement	102	1102		

Each data type has a unique data structure, such as the number of points or linear rings, followed by coordinates in 64-bit double numbers.

For example, the geometry `POINT(2.0 4.0)` is represented as: `0x0000000140000000000000004010000000000000`, where:

- 1-byte integer: `00` or 0: big endian
- 4-byte integer: `00000001` or 1: POINT (2D)
- 8-byte float: `4000000000000000` or 2.0: *x*-coordinate
- 8-byte float: `4010000000000000` or 4.0: *y*-coordinate

Format variations [edit]

EWKT and EWKB – Extended Well-Known Text/Binary

A PostGIS-specific format that includes the spatial reference system identifier (SRID) and up to 4 ordinate values (XYZM).[4][5] For example: `SRID=4326;POINT(-44.3 60.1)` to locate a longitude/latitude coordinate using the WGS 84 reference coordinate system.

***AGF Text* – Autodesk Geometry Format**

An extension to OGC's Standard (at the time), to include curved elements; most notably used in MapGuide.[6]

Software support [edit]

Database engines [edit]

- Amazon Redshift since November 21, 2019[7]
- Apache Drill supports full range of geospatial queries since version 1.12 as well as reading ESRI Shape files (SHP).
- Apache Solr enterprise search server since 4.0[8] through JTS
- Elasticsearch distributed, RESTful search and analytics engine since 6.2[9]
- Google BigQuery supports BigQuery Geographic Information Systems since April 2019.
- GigaSpaces InsightEdge[10]
- PostgreSQL with PostGIS Module 2.0
- Kinetica GPU-accelerated geospatial database
- Oracle Spatial 9i, 10g, 11g
- OmniSci since 4.0
- MarkLogic Server since 4.2[11]
- MemSQL since 4[12]
- MySQL since 4.1[13]
- MariaDB, all versions
- Neo4J[14]
- OrientDB[15]
- IBM DB2 LUW 9, 10 with Spatial Extender
- IBM DB2 for z/OS 9, 10 with Spatial Support
- IBM Netezza with Netezza Spatial
- IBM Informix 9,10,11 with Spatial datablade module
- Microsoft SQL Server since 2008 R2[16]
- SpatiaLite
- Teradata 6.1, 6.2, 12, 13 (native in 13 through add-in in previous versions)
- Ingres GeoSpatial
- Altibase 5.x
- SQL Anywhere 12
- SAP HANA SPO7,SP08
- H2 since 1.3.173 (2013-07-28)[17]
- Vertica since 7.1.0[18]
- VoltDB since V6.0[19]

APIs [edit]

- Boost C++ libraries (C++): See Geometry io/wkt and headers
- Esri geometry-api-java
- GEOS (C/C++)
  - Shapely (Python): See Shapely Documentation and Shapely in PyPI
  - GeoPHP (PHP)
- GDAL (C/C++ with bindings to Java, Python, and others)
- GeoRust: rust-wkt (Rust bindings)
- JTS Topology Suite (Java)
- SpatialKit (Java)
- NetTopologySuite (.NET)
- OpenLayers (JavaScript)
- OpenScales (ActionScript)
- parsewkt (Python) is a peg parser from WKT to python dictionaries
- pygeoif (Python) parses wkt with regular expressions
- rgeos (Ruby)
- sfd (R)
- Terraformer (JavaScript)
- WellKnownLib (C#.Net) Well-Known Text and Binary Parser

Protocols [edit]

- GeoSPARQL
- SensorThings API

See also [edit]

- Simple Features
- Geography Markup Language
- Well-known text representation of coordinate reference systems

References [edit]

1. ^ Herring, John R., ed. (2011-05-28), *OpenGIS® Implementation Standard for Geographic information – Simple feature access – Part 1: Common architecture*, Open Geospatial Consortium, retrieved 2019-01-28
2. ^ *Information technology – Database languages – SQL multimedia and application packages – Part 3: Spatial*) (5th ed.), ISO, 2016-01-15, retrieved 2019-01-28
3. ^ See the OGC Implementation Specification for geographic information – Simple Feature Access, section 6.1.11.1 , http://www.opengeospatial.org/standards/sfa
4. ^ https://github.com/postgis/postgis/blob/2.1.0/doc/ZMSgeoms.txt
5. ^ http://postgis.org/docs/ST_GeomFromEWKT.html
6. ^ http://e-logistics-plans.gdfsuez.com/mapguide/help/webapi/da/dd3/group___agf_text.html
7. ^ Amazon Redshift announces support for spatial data
8. ^ Solr GEO support
9. ^ Well-Known Text (WKT) Input Type in Elasticsearch documentation
10. ^ "Geospatial Queries". *docs.gigaspaces.com*. Retrieved 2020-06-02.
11. ^ https://docs.marklogic.com/guide/search-dev/geospatial
12. ^ http://docs.memsql.com/docs/geospatial-guide
13. ^ Well-Known Text (WKT) Format, MySQL documentation
14. ^ https://neo4j-contrib.github.io/spatial/
15. ^ https://orientdb.com/docs/3.1.x/indexing/Spatial-Index.html
16. ^ https://docs.microsoft.com/en-us/previous-versions/sql/sql-server-2008-r2/bb933970(v=sql.105)
17. ^ H2 create spatial index documentation
18. ^ "HP Vertica 7.1.x Release Notes". *my.vertica.com*. Retrieved 2018-03-21.
19. ^ https://www.voltdb.com/company/press-releases/voltdb-adds-geospatial-query-support-industrys-innovative-fast-data-platform/

External links [edit]

- Simple Feature Access Specification
- ISO Spatial standard (there is a charge for this)
- BNF Notation of WKT
- EBNF Notation of WKT
- Online conversion between geometry objects representations

Categories: GIS file formats | Markup languages