Spatial Database for Geographic Information System (GIS) Applications

Lect.dr.eng. Alexandru Boicea
Politehnica University of Bucharest, Faculty of Automation and Computer Science
Splaiul Independentei Street , No. 313, Bucharest 6, Romania
E-mail: alexb@cs.pub.ro

Abstract

This paper explains the concepts of Oracle Spatial as a way to store and analyze spatial data. Oracle Spatial is designed to make spatial data management easier and more natural to users of location-enabled applications and Geographic Information System(GIS) applications. Once spatial data is stored in an Oracle database, it can be easily manipulated, retrieved, and related to all other data stored in the database. The data that indicates the Earth location (such as longitude and latitude) of these rendered objects is the spatial data. When the map is rendered, this spatial data is used to project the locations of the objects on a two-dimensional piece of paper. A GIS is often used to store, retrieve, and render this Earth-relative spatial data. Types of spatial data (other than GIS data) that can be stored using Oracle Spatial include data from Computer Aided Design (CAD) and Computer Aided Manufacturing (CAM) systems.

1. Introduction

Classical database techniques used for spatial data X and Y columns from various geometries, but this method allows only limited indexing and querying and the database access and data manipulation is difficult.
Oracle Spatial in Oracle Database10*g* includes native data types for storing vector, raster, and persistent topology data. This document outlines some best practices when working with Oracle's native vector data type, SDO_GEOMETRY, in Oracle Spatial for Oracle Database 9*i* and 10*g*.
Some of the major GIS vendors have legacy proprietary alternatives to store vector data in Oracle, for example, in Oracle's LONG RAW data type. These legacy alternatives were introduced prior to the availability of the SDO_GEOMETRY data type. In the marketplace, there is sometimes the misconception that using SDO_GEOMETRY as the underlying vector data store may compromise features or performance in products offered by the major GIS vendors.
Oracle Spatial performance may far exceed 15% better performance over proprietary "LONG RAW" data type solutions[6]. This is because Oracle does not support table partitioning on tables that contain LONG or LONG RAW columns. Tables with SDO_GEOMETRY columns can leverage Oracle table partitioning, which can significantly help performance, scalability, and manageability.
Most business information has a location component, such as geocoded customer addresses, sales territories, and wireless service boundaries. Businesses can take advantage of their geographic information by incorporating location analysis into their information systems.
.

2. Oracle Spatial

Oracle Spatial provides a SQL schema and functions that facilitate the storage, retrieval, update, and query of collections of spatial features in an Oracle database. Spatial consists of the following[3]:

- A schema (MDSYS) that prescribes the storage, syntax, and semantics of supported geometric data types
- A spatial indexing mechanism
- Operators, functions, and procedures for performing area-of-interest queries, spatial join queries, and other spatial analysis operations
- Functions and procedures for utility and tuning operations
- Topology data model for working with data about nodes, edges, and faces in a topology Network data model for representing capabilities or objects that are modeled as nodes and links in a network
- GeoRaster, a feature that lets you store, index, query, analyze, and deliver GeoRaster data, that is, raster image and gridded data and its associated metadata
- The spatial component of a spatial feature is the geometric representation of its shape in some coordinate space. This is referred to as its geometry.

3. Object Relational Model

Spatial supports the *object relational model* for representing geometries. This model stores an entire

geometry in the Oracle native spatial data type for vector data, SDO_GEOMETRY. An Oracle table can contain one or more SDO_GEOMETRY columns. The object-relational model corresponds to a "SQL with Geometry Types" implementation of spatial feature tables in the Open GIS ODBC/SQL specification for geospatial features. The benefits provided by the object-relational model include:

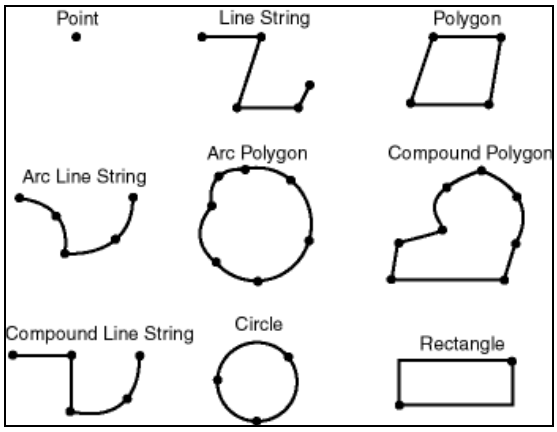
- Support for many geometry types, including arcs, circles, compound polygons, compound line strings, and optimized rectangles
- Ease of use in creating and maintaining indexes and in performing spatial queries
- Index maintenance by the Oracle database
- Geometries modeled in a single column
- Optimal performance

4. Geometry Types

A geometry is an ordered sequence of vertices that are connected by straight line segments or circular arcs. The semantics of the geometry are determined by its type. Spatial supports several primitive types, and geometries composed of collections of these types, including two-dimensional[7]:

- Points and point clusters
- Line strings
- n-point polygons
- Arc line strings (circular arcs.)
- Arc polygons
- Compound polygons
- Compound line strings
- Circles
- Optimized rectangles

Self-crossing polygons are not supported, although self-crossing line strings are supported. If a line string crosses itself, it does not become a polygon. A self-crossing line string does not have any implied area. *Two-dimensional points* are elements composed of two ordinates, X and Y, often corresponding to longitude and latitude. *Line strings* are composed of one or more pairs of points that define line segments. *Polygons* are composed of connected line strings that form a closed ring, and the area of the polygon is implied. The next figure illustrates the geometric types:



Examples:

- Points can represent: buildings, fire hydrants, utility poles, oil rigs, boxcars, or roaming vehicles.
- Lines can represent: roads, railroad lines, utility lines, or fault lines.
- Polygons can represent: cities, districts, flood plains, or oil and gas fields. A polygon with a hole might geographically represent a parcel of land surrounding a patch of wetlands.

5. Data Model

The Spatial data model is a hierarchical structure consisting of elements, geometries, and layers. Layers are composed of geometries, which in turn are made up of elements.

5.1 Element

An *element* is the basic building block of a geometry. The supported spatial element types are points, line strings, and polygons. For example, elements might model star constellations (point clusters), roads (line strings), and county boundaries (polygons). Each coordinate in an element is stored as an X,Y pair. The exterior ring and the interior ring of a polygon with holes are considered as two distinct elements that together make up a complex polygon. *Point data* consists of one coordinate. *Line data* consists of two coordinates representing a line segment of the element. *Polygon data* consists of coordinate pair values, one vertex pair for each line segment of the polygon. Coordinates are defined in order around the polygon (counterclockwise for an exterior polygon ring, clockwise for an interior polygon ring).

5.2 Geometry

A *geometry* (or *geometry object*) is the representation of a spatial feature, modeled as an ordered set of primitive elements. A geometry can consist of a single element, which is an instance of one of the supported primitive types, or a homogeneous or heterogeneous collection of elements. A multipolygon, such as one used to represent a set of islands, is a homogeneous collection. A heterogeneous collection is one in which the elements are of different types, for example, a point and a polygon. An example of a geometry might describe the buildable land in a town. This could be represented as a polygon with holes where water or zoning prevents construction.

5.3 Layer

A *layer* is a collection of geometries having the same attribute set. For example, one layer in a GIS might include topographical features, while another describes population density, and a third describes the network of roads and bridges in the area (lines and points). The geometries and associated spatial index for each layer are stored in the database in standard tables.

5.4 Coordinate System

A *coordinate system* (also called a *spatial reference system*) is a means of assigning coordinates to a location and establishing relationships between sets of such coordinates. It enables the interpretation of a set of coordinates as a representation of a position in a real world space. Any spatial data has a coordinate system associated with it. The coordinate system can be *georeferenced* (related to a specific representation of the Earth) or not georeferenced (that is, Cartesian, and not related to a specific representation of the Earth). If the coordinate system is georeferenced, it has a default *unit of measurement* (such as meters) associated with it, but you can have Spatial automatically return results in another specified unit (such as miles). Spatial data can be associated with a Cartesian, geodetic (geographical), projected, or local coordinate system:

- *Cartesian coordinates* are coordinates that measure the position of a point from a defined origin along axes that are perpendicular in the represented two-dimensional or three-dimensional space. If a coordinate system is not explicitly associated with a geometry, a Cartesian coordinate system is assumed.
- *Geodetic coordinates* (sometimes called geographic coordinates) are angular coordinates (longitude and latitude), closely related to spherical polar coordinates, and are defined relative to a particular Earth geodetic datum. (A geodetic datum is a means of representing the figure of the Earth and is the reference for the system of geodetic coordinates).
- *Projected coordinates* are planar Cartesian coordinates that result from performing a mathematical mapping from a point on the Earth's surface to a plane. There are many such mathematical mappings, each used for a particular purpose.
- *Local coordinates* are Cartesian coordinates in a non-Earth (non-georeferenced) coordinate system. Local coordinate systems are often used for CAD applications and local surveys.

When performing operations on geometries, Spatial uses either a Cartesian or curvilinear computational model, as appropriate for the coordinate system associated with the spatial data.

6. Database Objects

Oracle Spatial can store and analyze four-dimensional data; however, for the sake of clarity, this article discusses only two-dimensional. The basic building block in Oracle Spatial is an element: a Point, a LineString, or a Polygon.

Examples for each element are below:

Element	Name	Ordinates (Vertices Array)
	Point	{ 1,2 }
	LineString	{ 2,2, 5,1, 3,0 }
	Polygon	{ 0,0, 1,0, 1,1, 0,1, 0,0 }

Oracle Spatial defaults to a Cartesian coordinate system with dimensionless units. For large geodetic data sets (such as maps of the earth's surface), a geodetic coordinate system is needed. This coordinate system uses longitude and latitude for the ordinates and automatically handles issues with the curvature of the earth's surface. Another piece of information needed to map is the accuracy of ordinates. Spatial data almost always has an

associated tolerance or error associated with it. However, Oracle Spatial defaults to a tolerance of zero[5]. With a coordinate system and tolerance, it's possible to query and map spatial data correctly.
To create a table called **region** with an indexed spatial column, the following SQL statements are needed:

```
CREATE TABLE region (
   Id_region NUMBER PRIMARY KEY,
   name      VARCHAR2(32),
   shape     MDSYS.SDO_GEOMETRY);

INSERT INTO user_sdo_geom_metadata VALUES
  ( 'region',  'shape',
   MDSYS.SDO_DIM_ARRAY (
   MDSYS.SDO_DIM_ELEMENT('X', 0, 100, 0.05),
   MDSYS.SDO_DIM_ELEMENT('Y', 0, 100, 0.05)
   ),  NULL);

CREATE INDEX region_idx ON region (shape)
INDEXTYPE IS MDSYS.SPATIAL_INDEX;
```

The first statement creates the desired table with a column named shape to hold the spatial data. We'll discuss the MDSYS.SDO_GEOMETRY data type later.
The second statement tells Oracle Spatial about the spatial data in the region table. The user_sdo_geom_metadata table describes the coordinate system and tolerance for each spatial column that a user owns. The table takes four values: the table name that has a spatial column, the column name for the spatial data, an array describing the minimum, maximum, and tolerance value for each dimension, and a number stating the coordinate system. region uses two-dimensional data that can range from 0 to 100 units with a tolerance of 0.05.
The last statement creates the spatial index, which is required for spatial queries. Oracle uses two types of spatial indexing: R-tree and Quadtree. If the spatial data is geodetic (map data), R-Tree must be used to take full advantage of Oracle Spatial's functions. In the index statement above the spatial type wasn't specified, causing Oracle Spatial to default to an R-Tree index. Having created an indexed table to store spatial data, all that's left is to populate it with data.
Oracle Spatial uses the MDSYS.SDO_GEOMETRY type to store spatial data, which is defined as:

```
CREATE TYPE sdo_geometry AS OBJECT (
   SDO_GTYPE NUMBER,
   SDO_SRID NUMBER,
   SDO_POINT SDO_POINT_TYPE,
   SDO_ELEM_INFO
        MDSYS.SDO_ELEM_INFO_ARRAY,
   SDO_ORDINATES
        MDSYS.SDO_ORDINATE_ARRAY);
```

SDO_GTYPE is a number that defines the overall shape; that is, a line, a sphere, and so forth[6]. It describes the end result of the ordered combination of elements. SDO_GTYPE is a four-digit integer. The first digit represents the number of dimensions. The second digit represents the linear representation, which is important for a three- or four-dimensional shape. In a two-dimensional shape the value is zero. The last two digits represent the shape and range from 00 to 07.
Each value describes a particular geometry, noted below:

Val	Geometry	Description
00	UNKNOWN_ GEOMETRY	Spatial ignores this value
01	POINT	A single point element
02	LINE or CURVE	Contains one line string element that may be linear, curved or both
03	POLYGON	Contains one polygon element with or without other polygon elements in it
04	COLLECTION	A heterogeneous collection of elements
05	MULTIPOINT	Contains one or more points
06	MULTILINE or MULTICURVE	Contains one or more line string elements
07	MULTI POLYGON	Contains multiple polygon elements that maybe disjoint

Examples of SDO_GTYPE are a rectangle, 2003, and a collection of roadways, 2006.
The SDO_SRID number describes the coordinate system to use.
An SDO_POINT is defined as:

```
CREATE TYPE sdo_point_type AS OBJECT
        ( X NUMBER, Y NUMBER,  Z NUMBER);
```

The SDO_ORDINATES_ARRAY is a list of all the vertices that define the geometry. It's defined below:

```
CREATE TYPE sdo_ordinate_array AS VARRAY
(1048576) of  NUMBER;
```

The SDO_ELEM_INFO_ARRAY describes the multiple elements within the SDO_ORDINATES_ARRAY and is defined below:

```
CREATE TYPE sdo_elem_info_array AS VARRAY
(1048576) of  NUMBER;
```

An SDO_ELEM_INFO_ARRAY is understood as three values at a time. Each set of three values describes an element of the geometry. The region example is composed of two polygon elements and would have an SDO_ELEM_INFO_ARRAY containing six numbers: { 1,1003,1, 6, 2003, 1 }.

Value	Meaning
1	Point element
2	LineString element
1003	Exterior polygon element
2003	Interior polygon element

The SQL needed to insert a region into the region table is listed below:

```
INSERT INTO region
VALUES( 10, 'Prahova',
   MDSYS.SDO_GEOMETRY(
                              2003, NULL, NULL,
   MDSYS.SDO_ELEM_INFO_ARRAY
                             (1,1003,1, 19,2003,1),
   MDSYS.SDO_ORDINATE_ARRAY
(0,0,10,0, 10, 10,0,10, 0,0, 4,4, 6,4, 6,6, 4,6, 4,4)));
```

The first part of the query uses an Oracle Spatial function called SDO_FILTER, which is defined below:

```
SDO_FILTER(
geometry1   MDSYS.SDO_GEOMETRY,
geometry2   MDSYS.SDO_GEOMETRY,
params VARCHAR2)
```

The first geometry argument, geometry1, is a column name of spatially indexed geometries. The second geometry argument, geometry2, may or may not come from a table and does not need to be indexed spatially. The final argument defines how the filtering works. If the final argument is querytype=WINDOW, filtering is done in memory and works very well when geometry2 doesn't come from a table. If the final argument is querytype=JOIN, geometry2 must come from a table and performance depends on the spatial index type of the two tables. SDO_FILTER returns a string of TRUE if a successful filtering occurs or FALSE otherwise.
An example query using SDO_FILTER follows:

```
SELECT name  FROM region t
WHERE id_region = 12
AND SDO_FILTER (t.shape,
   mdsys.sdo_geometry (2003,NULL,NULL,
   mdsys.sdo_elem_info_array(1,1003,1),
   mdsys.sdo_ordinate_array(2,2, 5,2, 5,5, 2,5, 2,2)),
   'querytype=WINDOW') = 'TRUE';
```

This query selects all objects that have a geometry with an indexed grid square within the polygon defined.
To obtain an exact query, a second function called SDO_RELATE must be executed. SDO_RELATE looks at two geometries and determines if they interact in a specified way. It is important to note that SDO_RELATE only works on two-dimensional data. It is defined below:

```
SDO_RELATE(
geometry1 MDSYS.SDO_GEOMETRY,
geometry2 MDSYS.SDO_GEOMETRY,
params VARCHAR2)
```

SDO_RELATE arguments are the same as those for SDO_FILTER with the exception of the last argument. The params argument has a masktype value in addition to the querytype value.
To select all objects that are inside a defined rectangle the following query would work:

```
SELECT name  FROM region t
WHERE id_region = 12
AND SDO_FILTER (t.shape,
   mdsys.sdo_geometry(2003,NULL,NULL,
   mdsys.sdo_elem_info_array(1,1003,1),
   mdsys.sdo_ordinate_array(2,2, 5,2, 5,5, 2,5, 2,2)),
   'querytype=WINDOW') = 'TRUE'
AND SDO_RELATE(t.shape,
   mdsys.sdo_geometry(2003,NULL,NULL,
   mdsys.sdo_elem_info_array(1,1003,1),
   mdsys.sdo_ordinate_array(2,2, 5,2, 5,5, 2,5, 2,2)),
   'masktype=INSIDE
   querytype=WINDOW')='TRUE'
```

Oracle Spatial provides other powerful querying functions beside SDO_FILTER and SDO_RELATE:

Query / Functions	Description
SDO_NN	Nearest neighbor
SDO_SDO_WITHIN_DISTANCE	All geometries with a certain distance
SDO_GEOM.SDO_MBR	The minimum bounding rectangle for a geometry
SDO_GEOM.SDO_DISTANCE	The distance between two geometries
SDO_GEOM.SDO_INTERSECTION	Provides the intersection point of two geometries

There are many more query operators and functions in Oracle Spatial.

7. Conclusion

Oracle Spatial makes it possible to combine the relational power of a database with spatial data. The ability to use indexes, various queries, and functions means complex spatial calculations may be pushed back onto large database servers. As mobile applications and technologies increase, so will the demands to store and analyze spatial data in a transactional setting. Oracle Spatial are powerful core features of the Oracle database.

For complex GIS applications Oracle Spatial provides the required functionality, like spatial functions (including area, buffer, centroid calculations), advanced coordinate systems support, linear referencing systems, and aggregate functions.

Oracle Spatial provides performance, ease of use, and an architecture that scales to meet the needs of Internet and wireless solutions.

This technical document describes some best practices, tips and general information that can help utilize Oracle Spatial to increase productivity, decision support, and cost savings in your everyday business practices.

8. References

[1] B. Hall, *Oracle Spatial and Database Tuning*, Oklahoma, 2007

[2] R. Kothuri**,** A. Godfrind, *Pro Oracle Spatial for Oracle Database,* Hardcover 2007

[3] K.Loney, G.Kock, *Oracle Database 10G – The Complete Reference*, Oracle Press , Osborne 2004

[4] R. Kothuri, E. Beinat, A.Godfrind, *Pro Oracle Spatial,* Apress L. P. Hardcover, Nov. 2004

[5] M. Bauer, *Mapping Geometric Data with Oracle Spatial*, O'Reilly Network, Nov. 2003

[6] D. Geringer, *Oracle Spatial Best Practices,* Oracle Press , Osborne 2003

[7] www.oracle.com *Oracle® Spatial User's Guide and Reference10g,* Release 2 (10.2) Part Number B14255-03