Oracle Spatial, Geometries

Ravi Kothuri, Siva Ravada
Spatial Technologies, Oracle USA, Nashua, NH, USA

Synonyms

Oracle locator; Oracle spatial; Oracle geographic information system; Reference system, spatial; OGC, OGC simple features; SQL/MM; ISO/IEC; Minimum bounding rectangles; Indexing, spatial; Populating, topology

Definition

A geometry in Oracle Spatial can be one of the following[1]:

- Point
- Linestring connecting two or more points
- Polygon specified as a closed linestring and indicating an area bounded by the linestring (with zero or more holes inside the outer bounding linestring)
- Multipoint collection
- Multiline collection consisting of unconnected linestrings
- Multipolygon collection consisting of nonoverlapping polygons, or
- Heterogeneous collection consisting of a mixture of point, linestring, and/or polygon geometries

Each geometry can be specified in the context of a *spatial reference system*, also referred to as a *coordinate system*. Typical coordinate systems are *local* or *geographic*. Geographic coordinate systems are *geodetic* or *projected*. Whereas local and projected coordinate systems operate in an Euclidean space, geodetic coordinate systems operate on the Earth's surface.

Geometry data in Oracle Spatial can be stored using (1) the SDO_GEOMETRY data type or using (2) the SDO_TOPO_GEOMETRY data type. Whereas the SDO_GEOMETRY type stores the geometry as is, the SDO_TOPO_GEOMETRY stores it using simple topological primitive elements such as nodes, edges and faces. The latter representation allows for shared representation of edges (and other elements) between different feature geometries. For instance, a line (or edge) can be shared between the boundary of a land parcel as well as a road segment. The advantages of such shared representation are (1) the ease of topology management across multiple features: changes in the nodes, edges and faces automatically reflect in the higher-level geometry features; (2) elimination of redundancy translating into less storage space; and (3) sometimes faster processing for topological queries.

[1]Additional types to model geometric shapes in three-dimensional space are being added in subsequent releases of Oracle. This entry focuses only on two-dimensional geometries.

Historical Background

For several decades, defining, storing and manipulating geometry data has received much attention in various research communities including geographic information systems (GIS), computer-aided design (CAD)/ computer-aided manufacturing CAM, medical and virtual reality. Whereas the GIS industry deals with mapping points, lines and polygons on the surface of the earth using two-dimensional (2D) coordinates, the CAD/CAM industry deals with representing buildings, machines and other manmade artifacts usually in mostly three-dimensional (3D) spaces. The medical and virtual reality applications use a combination of 2D and 3D objects. The formats for representing the geometry data have varied across these fields, and also among different vendors in each of these fields. For instance, the spatial vendors such as Oracle Spatial, ESRI, Mapinfo, Autodesk, Bentley, and Intergraph each have their own formats to store 2D geometry data.

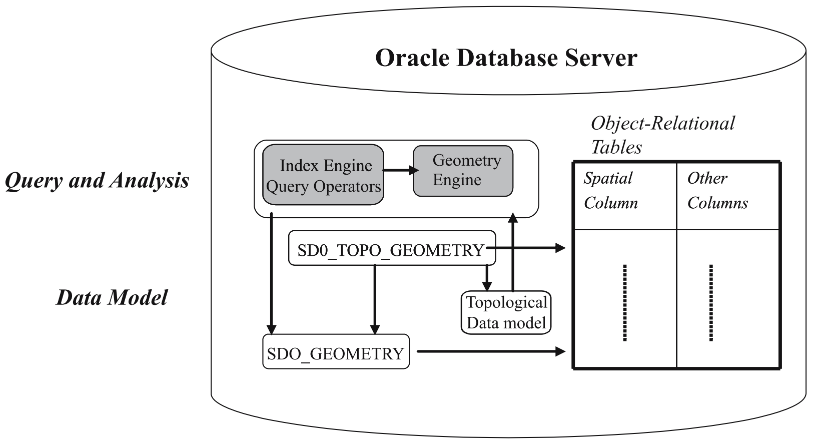
Over the past few decades, several commercial and research products for managing geometry data have emerged. The majority of the initial products either stored them in a file-based ASCII format or some vendor-specific proprietary binary formats (only understood by vendor tools). Storage in file systems has its limitations with regard to size, query/access, scalability, and recoverability. Storing in binary formats makes the data uninteroperable. To make the data across different vendors interoperable, the GIS community has defined two standards:

- The Open GIS Consortium (OGC) has the Simple Features specification[2] for storage, retrieval, query, and update of simple geospatial features. This specification defines a geometry type with appropriate subtypes to model points, line strings, polygons, and so on.
- Geometry data is explicitly dealt with in Part 3, of SQL/MM[3] (the ISO/IEC[4] international standard for "Text, Spatial, Still Images, and Data Mining"). This specification deals with spatial user-defined types and associated routines to store, manage, and retrieve spatial data. This standard specifies the ST_Geometry type to store spatial data. This type has subtypes such as ST_Point, ST_LineString, and ST_Polygon to mod-

[2]Open GIS Consortium, *OpenGIS Simple Features Specification for SQL Revision 1.1*, http://www.opengis.org/docs/99-049.pdf, May 5, 1999.

[3]Structured Query Language (Multimedia)

[4]International Organization for Standardization/International Electrotechnical Commission.



Oracle Spatial, Geometries, Figure 1 Geometries in Oracle Spatial

el different types of spatial geometries. This standard also specifies a well-known text format for specifying geometries. For instance, the string "POINT(1 1)" indicates a point geometry with coordinates at (1,1). Note that the types represented in OGC Simple Features are a subset of those defined by SQL/MM Part 3.

Oracle Spatial's model for storing geometry data is an open format, i. e., a nonbinary representation that can be read by any vendor. Furthermore, the data is stored as first-class objects inside an Oracle database. This extends all the salient features of a database such as atomicity, concurrency, scalability and recoverability to geometry data. Users can query spatial and nonspatial data in a single query. In addition to storing the geometry data as an open format (i. e., a format that can be read/understood by nonOracle software), Oracle Spatial is also working on defining web services on top of the open SDO_GEOMETRY type to interoperate with other GIS applications.

Scientific Fundamentals

Figure 1 shows the architecture for geometry storage and manipulation in Oracle Spatial. The data model has two main data types, SDO_GEOMETRY and SDO_TOPO_GEOMETRY, to store the geometry data. These types can be stored as columns of a table in Oracle DB. After storing geometry data in Oracle tables, users can operate on them using appropriate spatial query operators and functions. These operators are processed in a multistep refinement process: for instance, using a spatial index engine and then a geometry engine (see [1] for more details) when the data are stored as SDO_GEOMETRY objects.

Figure 2 shows a sample of typical geometry data in GIS applications: a polygon geometry representing the "city hall" property boundary and an adjoining street "City Hall St". An illustration of how to store and manipulate this data using the two alternative models for storing geometries in Oracle Spatial – the SDO_GEOMETRY and the SDO_TOPO_GEOMETRY follows.

The SDO_GEOMETRY Model

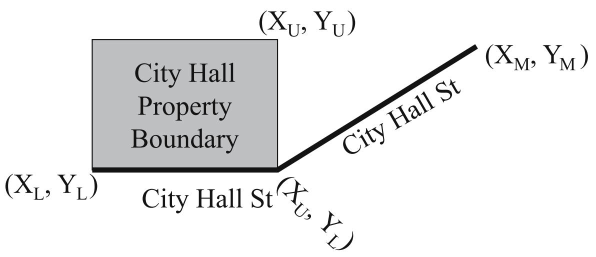
Storage To store properties and streets in the SDO_GEOMETRY model, spatial users can create tables of the form:

```
CREATE TABLE city_properties
(property_name VARCHAR2(32),
   property_boundary SDO_GEOMETRY);
CREATE TABLE city_streets
(street_name VARCHAR2(32),
   street_geometry SDO_GEOMETRY);
```

Geometry data can be inserted into these tables as follows:

```
INSERT INTO city_properties
('CITY HALL',
        SDO_GEOMETRY(2003, 8307, null,
        SDO_ELEM_INFO_ARRAY(1,1003,3),
        SDO_ORDINATES(XL, YL, XU, YU))
);
```

Example 1 SQL for inserting a geometry into the city_ properties table



Oracle Spatial, Geometries, Figure 2 Typical geometry data

Observe that the second column instantiates an SDO_GEOMETRY object and inserts that into the property_boundary column of the city_properties table. The SDO_GEOMETRY object has the following fields (exact details in [2]:

- The SDO_GTYPE attribute specifies which type of shape (point, line, polygon, collection, multipoint, multiline, or multipolygon) that the geometry actually represents. In Example 1, the SDO_GTYPE is set to 2003, where the "2" indicates the dimensionality and the last digit "3" indicates it is a polygon geometry.
- The SDO_SRID attribute specifies the ID of the spatial reference system (coordinate system) in which the location/shape of the geometry is specified. In Example 1 this is set to 8307.
- If the geometry is a point (e. g., the location of customers), then users can store the coordinates in the SDO_POINT attribute of the SDO_GEOMETRY. Since the "city hall" geometry is not a point, this field is set to NULL.
- If the geometry is an arbitrary shape (e. g., a street network or city boundaries), then users can store the coordinates using the SDO_ORDINATES array and SDO_ELEM_INFO array attributes.
  - ORDINATES attribute stores the coordinates of all elements of the geometry.
  - The SDO_ELEM_INFO attribute specifies where in the SDO_ORDINATES array a new element starts, how it is connected (by straight lines or arcs), and whether it is a point (although the SDO_POINT is recommended for storage and performance reasons as explained below), a line, or a polygon. In the above example, the SDO_ELEM_INFO is set to (1,1003,3) indicating a rectangle and the lower vertex (X_L, Y_L) and upper vertex (X_U, Y_U) coordinates being specified in the SDO_ORDINATES array.

Other simple examples are point and line geometries. For instance, users can insert a a line geometry to represent a street adjoining the "city hall" as follows:

```
INSERT INTO city_streets
("CITY HALL ST",
        SDO_GEOMETRY(2002, 8307, null,
        SDO_ELEM_INFO_ARRAY(1,2,1),
        SDO_ORDINATE_ARRAY(XL,YL,
                XU,YL, XM,YM))
);
```

Example 2 SQL for inserting a geometry into the city_streets table

Note that the geometries for both "CITY HALL" and "CITY HALL ST" share the edge <(X_L, Y_L), (X_U, Y_L)> and is stored in both of them. If for some reason, the street boundary is modified to go inward into the city hall property, both geometries need to be updated. The SDO_TOPO_GEOMETRY model, discussed below, eliminates the need for such redundant storage and multiple updates.

Note that if the geometry is just a point, the SDO_GTYPE will be 2001 and the coordinates are stored in the SDO_POINT attribute. More detailed examples (such as multipoints, multilines, multipolygons and collections) can be found in Table 4-1 of [2].

Spatial Functions After storing geometries in the city_properties table, users can perform a variety of spatial operations including:

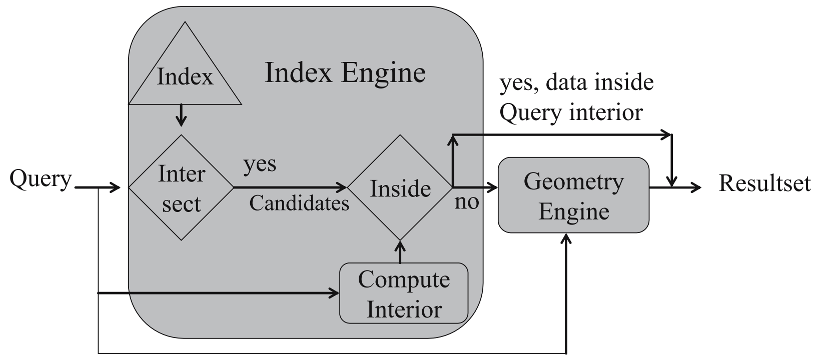
- SDO_GEOM.Relate: this function can determine the relationship of one geometry with another geometry. For instance, users can identify all city_properties that touch a query region r as follows:

```
SELECT * FROM city_properties
WHERE SDO_GEOM.RELATE
      (property_boundary, r, "TOUCH",
<tolerance_value>)="TRUE"
```

- SDO_GEOM.SDO_BUFFER: this function returns a buffer around a specified geometry. Users can use this function to determine the geometry that defines say, the 2-mile, radius around "city hall".
- SDO_GEOM.SDO_UNION, SDO_GEOM.intersection, . . . : these functions allow users to compute the union, or intersection of a pair of geometries.

Spatial Indexing and Query Processing Identifying the properties that touch the specified query using the SDO_GEOM.RELATE function can be slow, especially when the city_properties table has a large number of rows. A better approach is to use an index to prune the search. Oracle Spatial uses a variant of R*-trees (see [3] for details) and several other optimizations [1,5,6] for indexing spatial data. The spatial index stores approximations such asminimum bounding rectangles (MBRs) for each geometry. Queries are then compared with the MBR approximations first, and only appropriate candidates are passed on to an exact geometry–geometry comparison in the geometry engine (using the SDO_GEOM.RELATE and other spatial functions). This multistep processing filter out a majority of irrelevant geometries. The architecture for query processing is shown in Fig. 3.

Oracle Spatial has introduced additional optimizations such as those described in [1]. For each query, an interior component is identified. Any data geometries that are completely inside the interior of the query are included in the



Oracle Spatial, Geometries, Figure 3 Query processing on SDO_GEOMETRY data in Oracle

result set, bypassing the expensive geometry-engine processing. It has been show that this optimization is quite effective for large query windows and obtains substantial query speed increases [1].

The SDO_TOPO_GEOMETRY Model

In Examples 1 and 2 (Fig. 2), observe that the edge <(X_L, Y_L), (X_U, Y_L)> is shared between two SDO_GEOMETRY objects but stored in both. Such redundant storage not only increases storage requirements, but also necessitates *explicit* maintenance of the integrity between multiple geometry tables. For consistency with Oracle documentation, the geometry tables are hereafter referred to as *feature tables* and geometries in these tables as *features*.

In the SDO_TOPO_GEOMETRY or simply "topology" model, the city_properties and city_streets are created as follows:

```
Create Table City_Properties (
Feature_Name VARCHAR2(30) PRIMARY KEY,
Feature SDO_TOPO_GEOMETRY);
```

```
Create Table City_Streets (
Feature_Name VARCHAR2(30) PRIMARY KEY,
Feature SDO_TOPO_GEOMETRY );
```

Where the SDO_TOPO_GEOMETRY has the following structure:

```
TG_Type                     NUMBER
TG_ID                       NUMBER
TG_Layer_ID                 NUMBER
Topology_ID                 NUMBER
```

Together the set of feature layers constitute a "TOPOLOGY". For instance, users can create an explicit topology called "CITY MODEL" and add the feature layers "CITY_PROPERTIES" and "CITY_STREETS" to it.

When a TOPOLOGY is created, three dependent tables, <TOPOLOGY>_NODE$, <TOPOLOGY>_EDGE$, <TOPOLOGY>_FACE$, (or NODE$, EDGE$, and FACE$ for short) are created by Oracle Spatial. These tables respectively store the node-, edge- and face-type primitive elements of the specified topology.

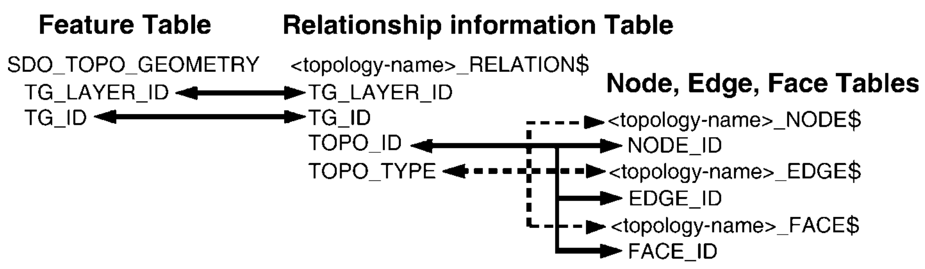
Each SDO_TOPO_GEOMETRY maps to primitive elements (that compose the feature) in the NODE$, EDGE$, and FACE$ table. This mapping is maintained in another system-generated RELATION$ table of the following structure:

```
TG_Layer_ID           NUMBER
TG_ID                 NUMBER
Topo_ID               NUMBER
Topo_type             NUMBER
```

Note that the <TOPO_ID, TOPO_TYPE> refer to the ID, and the type (node, edge, face) of the primitive element in the feature. This is described in more detail in Fig. 4.

Populating a TOPOLOGY A simple way to populate a topology is by populating the NODE$, EDGE$, and FACE$ tables for primitive elements directly. For instance, the EDGE$ table could store the edge from (X_L, Y_L), to (X_U, Y_L) as TOPO_ID=1, TOPO_TYPE=2. This edge can be referred to during the construction of features as follows:

```
INSERT INTO City_Streets VALUES
("City Hal St",
   SDO_TOPO_GEOMETRY(
      <TOPOLOGYname>,
   2002, - LINE FEATURE
      2, - TOPOLOGY_FEATURE_LAYER ID
      SDO_TOPO_OBJECT_ARRAY(
      - Array of lowerlevel primitives
      SDO_TOPO_OBJECT(1, 2)
      - Refer to Lower-level primitive
      - i.e., the edge
      - <(XL, YL),(XU, YL)>
        SDO_TOPO_OBJECT (2,2)
                               - another edge
         )
      );
```



Oracle Spatial, Geometries, Figure 4 Relationship between SDO_TOPO_GEOMETRY feature and the corresponding primitives

Likewise, the features in the CITY_PROPERTIES layer can be constructed.

An alternative approach is to insert an SDO_GEOMETRY as a feature into a topology. The following SQL shows an example of how to insert the linestring SDO_GEOMETRY connecting (X_L, Y_L), (X_U, Y_L,) and (X_M, Y_M) for the "CITY HALL ST" feature directly into the City_Streets feature layer.

```
INSERT INTO City_Streets
   VALUES ("City Hal St",
      sdo_topo_map.create_feature
                          ("<TOPOLOGYname>,
   SDO_GEOMETRY(2002,null,null,
          sdo_elem_info_array(1,2,1),
   SDO_ORDINATE_ARRAY
           (XL, YL, XU, YL, Xm, Ym) );
```

Indexing and Querying on Features Observe that the SDO_TOPO_GEOMETRY object in the feature tables does not store the "geometry" explicitly but only *points* (through its feature_id) to the topological primitives in the RELATION$ table. If the exact geometry needs to be materialized, the GETGEOMETRY method in the SDO_TOPO_GEOMETRY type can be utilized. For instance, the following SQL returns the geometry as an SDO_GEOMETRY from the CITY_STREETS feature layer.

```
SELECT feature.GETGEOMETRY() FROM
city_streets;
```

Note that the above function is evaluated by obtaining the composing primitive elements for the SDO_TOPO_GEOMETRY. This will be slower than storing the entire geometry as is as in the SDO_GEOMETRY model.

In most queries, users are more interested in identifying features that intersect either a specified query region or another topological feature. This can be performed with the help of the SDO_TOPO_ANYINTERACT operator.

```
SELECT * FROM city_streets
WHERE SDO_TOPO_ANYINTERACT(feature,
         query_sdo_geometry)="TRUE";
```

Note that there is no need for any explicit index creation in a topology. The RELATION$ table acts as the mapping table between features and topological primitive elements and is already indexed internally by the spatial engine. Spatial indexes are only necessary on the NODE$, EDGE$ and FACE$ tables and are maintained internally by Oracle Spatial.

Updating and Editing Topology As mentioned earlier, the main feature of creating and maintaining a topology is to ensure consistency and integrity for shared geometries. This means instead of updating the features directly, the users can update only the shared boundaries. Such update of shared elements will implicitly be reflected in all the features that share those updated elements. Typical functions provided for updates include addition/deletion of nodes, and addition/deletion of edges.

Key Applications

The SDO_GEOMETRY type is used to store geometry data in a number of applications [2]. Some examples include:

- Cadastral and census databases: all companies need to manage the location and shape of different assets (facilities, property). Likewise, government agencies such as Census Bureau maintain the exact shapes of administrative, geographical boundaries of different entities such as cities, counties and states. Other agencies manage the precise extent of individual properties and/or other land, natural resources. For each of these applications, there is an explicit need for storing large numbers of spatial geometries. The SDO_GEOMETRY type is ideal for such applications.
- Routing and navigation applications: a common use of spatial databases is route computation. The routing application server can store the streets, and highways inside the database and return the closest or fastest routes between two endpoints specified by a user.
- Mapping applications: several applications exist to construct maps from SDO_GEOMETRY data stored inside an Oracle database. Such maps can be integrated with other services for explorative analysis or for enabling business intelligence.

- Location-based services: Wireless data services increasingly use location data to enrich the user experience and provide valuable services. Uses include personal navigation systems, friend finders, panic button, roadside emergency, location-based yellow page searches, and the like. Locations of such wireless users can be stored and managed using the SDO_GEOMETRY type inside a database.
- Utility (electric, telephony) applications: the location and shape of electric and telephone networks can be effectively managed using the SDO_GEOMETRY type inside an Oracle database. Additional functionality in Oracle Spatial such as Linear Referencing can be utilized to effectively traverse such utility networks.
- Location-analysis applications: location analysis is frequently used in a variety of industries including banking, finance and insurance. For example in an insurance application, the risk factors for an entity (such as a person or a car) partially depend on its location. Management of the location of different entities using SDO_GEOMETRY and performing clustering analysis to identify high risk clusters, or to classify entities into different risk classes is a typical application in the insurance industry.

The SDO_TOPO_GEOMETRY is used in land management applications in local, state and federal governments. This model is useful for maintaining the integrity of land parcels and other territorial boundary management applications. The US Census Bureau and the Ordnance Survey model the land parcels using the SDO_TOPO_GEOMETRY type.

Future Directions

To summarize, Oracle Spatial offers two models for storing geometry data: the SDO_GEOMETRY and the SDO_TOPO_GEOMETRY models. The SDO_TOPO_GEOMETRY model is well suited for update-intensive applications. The SDO_GEOMETRY model may perform better if the majority of the workload consists of queries that fetch the entire geometry.

The types of geometries stored and processed in the SDO_GEOMETRY and the SDO_TOPO_GEOMETRY models are currently restricted to being 2D in nature. In future releases, Oracle Spatial may include support for storing 3D geometries (e. g., 3D solids) that appear in 3D city models and other advanced GIS applications. In addition to storing vector geometries, Oracle Spatial also supports efficient storage and retrieval of raster data. See Cross References for more details.

Cross References

► Raster Data

Recommended Reading

1. Kothuri, R., Ravada, S.: Efficient processing of large spatial queries. In: Proceedings of the Symposium of Spatial and Temporal Databases, Redondo Beach, CA, 12–15 July 2001
2. Kothuri, R., Godfrind, A., Beinat, E: Pro Oracle Spatial. Apress, Berkeley, CA (2004)
3. Kothuri, R., Ravada, S., Sharma, J., Banerjee, J.: Indexing medium dimensionality data in Oracle. In: Proceedings of the ACM SIGMOD International Conference on Management of Data. Philadelphia, PA, 1–3 June 1999
4. Beckmann, N., Kriegel, H., Schneider, R., Seeger, B.: The R* tree: An Efficient and Robust Access method for points and rectangles. In: Proceedings of the ACM SIGMOD International Conference on Management of Data. Atlantic City, NY 23–25 May, 1990
5. Samet H.: The Design and Analysis of Spatial Data Structures. Addison-Wesley, Reading (1989)
6. Brinkoff, T., Kreigel, H.-P., Schneider, R., Seegar, B.: Multi-step processing of spatial joins. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Minneapolis, MN, 24–27 May 1994
7. An, N., Kothuri, R., Ravada, S.: Improving performance with bulk inserts in Oracle R-trees. In: Proceedings of the International Conference on Very Large Data Bases (VLDB), Berlin, Germany, 9–12 Sept 2003
8. Oracle. (2006) 10g Release 2 Spatial User's Guide and Reference. http://www.oracle.com/products/spatial. Accesed 2006
9. White, D.A., Jain, R.: Similary Indexing Algorithms and Performance. In: Proc. SPIE. 2670, pp. 62–73 (1996)

Oracle Spatial GeoRaster

► Oracle Spatial, Raster Data

Oracle Spatial, Raster Data

QINGYUN (JEFFREY) XIE
Server Technologies & Location-Based Spatial Products, Oracle USA, Nashua, NH, USA

Synonyms

Oracle Spatial GeoRaster; DEM; Image; Field data; SQL; Gridded data

Definition

Oracle Spatial supports raster data storage and management within the Oracle RDBMS (Relational Database Management System) server. This Oracle database feature is called GeoRaster, which lets you store, index, query, analyze, and deliver raster image and gridded data and its associated metadata. GeoRaster provides a native SQL (Structured Query Language) data type and an object-relational schema. More specifically, it provides the SDO_GEORASTER data type, which can be used to