Oracle Spatial – Geometry and GeoRaster

Karine Reis Ferreira – karine@dpi.inpe.br

CAP 349 – Bancos de Dados Geográficos  (09/08/2014)

Oracle Spatial

- Oracle Spatial is an integrated set of functions and procedures that enables spatial data (vector and raster) to be stored, accessed, and analyzed in an Oracle database.

- Comercial system

- Current version: 11g

Oracle Spatial provides

- A schema (MDSYS) that prescribes the storage, syntax, and semantics of geometric and raster data types
- A spatial indexing mechanism
- Operators, functions, and procedures for performing area-of-interest queries, spatial join queries, and other spatial analysis 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
Oracle Spatial – Vector Data

Geometry data types

- Point
- Line String
- Polygon
- Arc Line String
- Arc Polygon
- Compound Polygon
- Compound Line String
- Circle
- Rectangle
Tipos de dados geométricos.
Operadores e funções espaciais.
Métodos de Acesso Espacial:
- R-Tree e QuadTree

CREATE TYPE SDO_GEOMETRY AS OBJECT (
  SDO_GTYPE NUMBER,
  SDO_SRID NUMBER,
  SDO_POINT SDO_POINT_TYPE,
  SDO_ELEM_INFO SDO_ELEM_INFO_ARRAY,
  SDO_ORDINATES SDO_ORDINATE_ARRAY);

Oracle Spatial – Vector Data
Criação de tabelas com tipos de dados espaciais:

```
CREATE TABLE distritossp
(
    cod NUMBER(32),
    sigla VARCHAR(10),
    denominacao VARCHAR(50),
    spatial_data MDSYS.SDO_GEOMETRY

    PRIMARY KEY (cod)
);
```
## Oracle Spatial – Metadata Tables

### MDSYS.CS_SRS

<table>
<thead>
<tr>
<th>SC_NAME</th>
<th>VARCHAR2 (68)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRID</td>
<td>NUMBER (38)</td>
</tr>
<tr>
<td>AUTH_SRID</td>
<td>NUMBER (38)</td>
</tr>
<tr>
<td>AUTH_NAME</td>
<td>VARCHAR2 (256)</td>
</tr>
<tr>
<td>WKTEXT</td>
<td>VARCHAR2 (2046)</td>
</tr>
<tr>
<td>SC_BOUDS</td>
<td>SDO_GEOMETRY</td>
</tr>
</tbody>
</table>

### USER_SDO_GEOM_METADATA

<table>
<thead>
<tr>
<th>TABLE_NAME</th>
<th>VARCHAR2 (32)</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLUMN_NAME</td>
<td>VARCHAR2 (32)</td>
</tr>
<tr>
<td>DIMINFO</td>
<td>SDO_DIM_ARRAY</td>
</tr>
<tr>
<td>SRID</td>
<td>NUMBER</td>
</tr>
</tbody>
</table>

### USER_SDO_INDEX_INFO

<table>
<thead>
<tr>
<th>SDO_INDEX_OWNER</th>
<th>VARCHAR2 (32)</th>
</tr>
</thead>
<tbody>
<tr>
<td>INDEX_NAME</td>
<td>VARCHAR2 (32)</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>VARCHAR2 (32)</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>VARCHAR2 (32)</td>
</tr>
<tr>
<td>SDO_INDEX_TYPE</td>
<td>VARCHAR2 (32)</td>
</tr>
<tr>
<td>SDO_INDEX_TABLE</td>
<td>VARCHAR2 (32)</td>
</tr>
<tr>
<td>SDO_INDEX_STATUS</td>
<td>VARCHAR2 (32)</td>
</tr>
</tbody>
</table>
Inserindo dados em tabelas com tipos de dados espaciais:

```sql
INSERT INTO distritossp (cod, sigla, denominacao, spatial_data)
VALUES (1, 'VMR', 'VILA MARIA'
MDSYS.SDO_GEOMETRY(2003, NULL, NULL,
MDSYS.SDO_ELEM_INFO_ARRAY( 1, 1003, 1 ),
MDSYS.SDO_ORDINATE_ARRAY(6,10, 10,1, 14,10,
10,14, 6,10)))
```
Oracle Spatial – Examples

- Indexando uma coluna espacial (R-Tree):

  ```sql
  CREATE INDEX distritossp_IDX
  ON distritossp(Spatial_DATA)
  INDEXTYPE IS MDSYS.Spatial_INDEX
  ```

- Funções para trabalhar com os índices:

  ```sql
  SDO_TUNE.QUALITY_DEGRADATION
  ALTER INDEX REBUILD
  ```
Oracle Spatial – Spatial Query

- Operadores:
  - Usados na cláusula WHERE de uma consulta SQL
  - Utilizam indexação espacial

<table>
<thead>
<tr>
<th>Operadores</th>
<th>Descrição</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDO_FILTER</td>
<td>Implementa o primeiro filtro do modelo de consulta (baseado nos MBR)</td>
</tr>
<tr>
<td>SDO_RELATE (SDO_TOUCH, SDO_ON, SDO_INSIDE)</td>
<td>Avalia se as geometrias possuem uma determinada relação topológica</td>
</tr>
<tr>
<td>SDO_WITHIN_DISTANCE</td>
<td>Verifica se duas geometrias estão dentro de uma determinada distância.</td>
</tr>
<tr>
<td>SDO_NN</td>
<td>Identifica os n vizinhos mais próximos de uma geometria</td>
</tr>
</tbody>
</table>
Funções:

- Definidas como subprogramas PL/SQL
- Usados na cláusula WHERE ou em SUBCONSULTAS
- Podem ser utilizadas sobre colunas espaciais não indexadas

<table>
<thead>
<tr>
<th>Funções</th>
<th>Descrição</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDO_INTERSECTION, SDO_UNION</td>
<td>Operações de conjunto</td>
</tr>
<tr>
<td>SDO_DIFFERENCE, SDO_XOR</td>
<td></td>
</tr>
<tr>
<td>SDO_BUFFER, SDO_CENTROID,</td>
<td>Operações que geram novas</td>
</tr>
<tr>
<td>SDO_CONVEXHULL</td>
<td>geometrias</td>
</tr>
<tr>
<td>SDO_AREA, SDO_LENGTH,</td>
<td>Operações métricas</td>
</tr>
<tr>
<td>SDO_DISTANCE</td>
<td></td>
</tr>
</tbody>
</table>
Oracle Spatial – Spatial Query

Buffer

Tolerance

```
geom1  geom2
```

```
geom1  geom2
```

```
geom1  geom2
```

```
geom1  geom2
```

```
geom1  geom2
```

```
geom1  geom2
```
Oracle Spatial – Spatial Query

Topological relations

- A CONTAINS B
  - B INSIDE A
- A COVERS B
  - B COVEREDBY A
- A TOUCH B
  - B TOUCH A
- A OVERLAPBDYINTERSECT B
  - B OVERLAPBDYINTERSECT A
- A OVERLAPBDYDISJOINT B
  - B OVERLAPBDYDISJOINT A
- A EQUAL B
  - B EQUAL A
  - (2 polygons with identical coordinates)
- A DISJOINT B
  - B DISJOINT A
- B ON A
  - A COVERS B
“Recuperar o nome de todos os municípios da grande São Paulo que são vizinhos ao município de São Paulo”.

```
SELECT d2.nomemunicp
FROM   grande_sp d1,
        grande_sp d2
WHERE  SDO_TOUCH (d1.spatial_data,
                  d2.spatial_data) = 'TRUE'
AND    (d2.nomemunicp <> 'SAO PAULO')
AND    (d1.nomemunicp = 'SAO PAULO')
```
“Recuperar todos os distritos que estão num raio de 3Km de um determinado rio”

```sql
SELECT di.deno
FROM sp_distritos di,
     sp_drenagem dr,
     user_sdo_geom_metadata m,
WHERE SDO_RELATE (di.spatial_data,
                  SDO_BUFFER (dr.spatial_data, m.diminfo, 3000),
                  'mask=INSIDE+TOUCH+OVERLAPBDYINTERSECT') = 'TRUE'
AND m.table_name = 'sp_drenagem'
AND m.column_name = 'spatial_data'
AND dr.object_id = '59';
```
Oracle Spatial – Raster Data

- GeoRaster is a feature of Oracle Spatial that lets you store, index, query, analyze, and deliver **raster data** and its associated metadata.
Oracle GeoRaster – Architecture

Five components to support the storage and use of raster data in Oracle Database:
Oracle GeoRaster – Architecture

Five components to support the storage and use of raster data in Oracle Database:

- GeoRaster Engine: provides the native GeoRaster object type and GeoRaster functionality including raster data and metadata indexing, update, query and manipulations.
Oracle GeoRaster – Architecture

Five components to support the storage and use of raster data in Oracle Database:

SQL API: standard SQL access to the raster and grid-based data in GeoRaster databases.
Oracle GeoRaster – Architecture

Five components to support the storage and use of raster data in Oracle Database:

SQL API: standard SQL access to the raster and grid-based data in GeoRaster databases.

Three PL/SQL packages:
(1) MDSYS.SDO_GEOR: for creating, modifying, and retrieving GeoRaster objects
(2) MDSYS.SDO_GEOR_UTL: for utility operations related to GeoRaster
(3) MDSYS.SDO_GEOR_ADMIN: for administrative operations related to GeoRaster
Oracle GeoRaster – Architecture

Five components to support the storage and use of raster data in Oracle Database:

C/C++/Java – Java, OCI, and OCCI: access to the raster and grid based data in GeoRaster with or without calling the GeoRaster SQL API.
Oracle GeoRaster – Architecture

Five components to support the storage and use of raster data in Oracle Database:

Viewing Tools: A variety of third party visualization and analysis tools:
(1) Oracle Fusion Middleware MapViewer;
(2) GeoRaster Viewer: a standalone viewer comes with the Oracle GeoRaster installation and can be used as a development or DBA tool.
Oracle GeoRaster – Architecture

Five components to support the storage and use of raster data in Oracle Database:

Input and Output [data] adapters: Facilitate loading and unloading raster data between well-known image file formats and GeoRaster. A variety of third party ETL tools now support loading and unloading GeoRaster data. GeoRaster also provides limited importing and exporting capability on six standard image file formats through both the server-side SQL API and the client-side Java tool.
Oracle defines *georaster* object as a multidimensional matrix of cells (raster) and a set of metadata. It is logically layered.
GeoRaster – Logical Data Model

Raster: a **multidimensional** matrix of raster cells. Each cell is one element of the matrix, and its value is called the cell value. The matrix has a number of dimensions, a cell depth, and a size for each dimension.

It can be **blocked** for optimal storage, retrieval and processing.

**Pyramids** (generalized, lower-resolution versions of the image – useful for fast retrieval in web applications) of the core raster data can be generated, stored and processed the same way.
A georaster object is logically layered. The core data is called the object layer or layer 0, and consists of one or more logical layers (or sublayers).
**GeoRaster – Logical Data Model**

**GeoRaster metadata:**
1. Object information (cell depth, blocking size, compression, info about pyramids, …)
2. Raster information
3. Spatial reference system information
4. Date and time (temporal reference system) information
5. Spectral (band reference system) information
6. Layer information for each layer (RGB colormap, grayscale lookup table, statistics, NODATA values, value ranges, …)
GeoRaster Engine

Physically, the GeoRaster data model is embodied as:

1. two native data types: SDO_GEORASTER and SDO_RASTER
2. an object-relational schema inside Oracle ORDBMS.
GeoRaster – Database Schema

Schema designed to store and manage raster data inside the database.
GeoRaster – Database Schema

Schema designed to store and manage raster data inside the database.

GeoRaster table: A GeoRaster table is any user-defined table, which has at least one data column of type SDO_GEORASTER.
GeoRaster – Database Schema

Schema designed to store and manage raster data inside the database.

SDO_GEORASTER Object:
include metadata and information about how to retrieve the raster data stored in another user-defined table called a Raster Data Table.
GeoRaster – Database Schema

Schema designed to store and manage raster data inside the database.

Raster data table: user-defined table which is an object table of type SDO_RASTER.

SDO_RASTER Object: includes a BLOB column called RASTERBLOCK, which stores the real raster blocks.
GeoRaster – Database Schema

Schema designed to store and manage raster data inside the database.

Other information associated with the GeoRaster objects can be stored in separate columns or tables, such as a Value Attribute Table (VAT).
SDO_GEORASTER Object

Native data type: each image or raster grid is stored as a single object of this native type.

```sql
CREATE TYPE sdo_georaster AS OBJECT (  
rasterType NUMBER,  
spatialExtent SDO_GEOMETRY,  
rasterDataTable VARCHAR2(32),  
rasterID NUMBER,  
metadata XMLType);
```
SDO_GEORASTER Object

Native data type: each image or raster grid is stored as a single object of this native type.

```
SQL> describe mdsys.sdo_georaster
       Name                     Null? Type
------------------------------------------------
RASTERTYPE                    NUMBER
SPATIALEXTENT                MDSYS.SDO_GEOMETRY
RASTERDATATABLE              VARCHAR2(32)
RASTERID                     NUMBER
METADATA                      XMLTYPE
```
SDO_GEORASTER Object

Native data type: each image or raster grid is stored as a single object of this native type.

```
SQL> describe mdsys.sdo_georaster
Name                        Null?  Type
------------------------------------------
RASTERTYPE                  NUMBER  MDSYS.SDO_GEOMETRY
SPATIALEXTENT               VARCHAR2(32)
RASTERDATATABLE             NUMBER  VARCHAR2(32)
RASTERID                    XMLTYPE
METADATA                    XMLTYPE
```

- **RASTERTYPE**: contains dimensionality information and the data type that can be extended
- **SpatialExtent**: spatial extent of the raster. GeoRaster uses R-Tree to index them.
- **RasterDataTable**: the table name where the raster is physically stored.
- **RasterId**: the index of the raster in the *Raster Data Table*.
- **Metadata**: XML document (Oracle XML Type data type) according to the GeoRaster metadata XML schema defined by GeoRaster
5-digit number in the format \([d][b][t][gt]\), where:

\([d]\) identifies the number of spatial dimensions. Must be 2 for the current release.

\([b]\) indicates band or layer information: 0 means one band or layer; 1 means one or more than one band or layer.

\([t]\) is reserved for future use and should be specified as 0 (zero).

\([gt]\) identifies the 2-digit GeoRaster type:
00 Reserved for Oracle use.
01 Any GeoRaster type. This is the only value supported for the current release.
02-50 Reserved for Oracle use.
51-99 Reserved for customer use in future releases.
For example, a RasterType value of 20001 means:

Two-dimensional data
One band (layer)
Any GeoRaster type
SDO_RASTER Object

Native type: each block of the image or raster grid (of a SDO_GEORASTER object) is stored as a single object of this type.

```
CREATE TYPE sdo_raster AS OBJECT (  
rasterID NUMBER,  
pyramidLevel NUMBER,  
bandBlockNumber NUMBER,  
rowBlockNumber NUMBER,  
columnBlockNumber NUMBER,  
blockMBR SDO_GEOMETRY,  
rasterBlock BLOB);
```
**SDO_RASTER Object**

Native type: each block of the image or raster grid (of a SDO_GEORASTER object) is stored as a single object of this type.

```
SQL> describe mdsys.sdo_raster
Name          | Null? | Type               |
--------------|-------|--------------------|
RASTERID      |       | NUMBER             |
PYRAMIDLEVEL  |       | NUMBER             |
BANDBLOCKNUMBER|       | NUMBER             |
ROWBLOCKNUMBER|       | NUMBER             |
COLUMNBLOCKNUMBER|   | NUMBER             |
BLOCKMBR      |       | MDSYS.SDO_GEOMETRY |
RASTERBLOCK   |       | BLOB               |
```
SDO_RASTER Object

Native type: each block of the image or raster grid (of a SDO_GEORASTER object) is stored as a single object of this type.

```
SQL> describe mdsys.sdo_raster
Name          Null? Type
------------- -------- -----------------
RASTERID      NUMBER
PYRAMIDLEVEL  NUMBER
BANDBLOCKNUMBER NUMBER
ROWBLOCKNUMBER NUMBER
COLUMNBLOCKNUMBER NUMBER
BLOCKMBR      MDSYS.SDO_GEOMETRY
RASTERBLOCK   BLOB
```

- **RasterId**: the raster id
- **PyramidLevel**: the pyramid level of this block
- **BandBlockNumber**: the band block number
- **RowBlockNumber**: the row block number
- **ColumnBlockNumber**: the column block number
- **BlockMBR**: the precise extent of the block
- **RasterBlock**: the raster block as a binary large object (BLOB)
Example

CITY_IMAGES table
(one row per city)

For each row (each image):
(Various user-defined columns...)
SDO_GEORASTER object
(for example, for Boston)

SDO_GEORASTER object

GeoRaster type
Spatial extent for this image (SDO_GEOMETRY)
Raster data table name (table of SDO_RASTER)
Raster ID
Metadata (SYS.XMLType)

Raster data table
(one row of SDO_RASTER object type for each block)

For each row (each block of the image):
(Raster ID, pyramid level, ...)
MBR for this block (SDO_GEOMETRY)
Image data for this block (BLOB)
In GeoRaster, *band* and *layer* are different concepts.

**Band** is a physical dimension of the multidimensional raster data set. Bands are numbered from 0 to \( n-1 \), where \( n \) is the highest layer number.

**Layer** is a logical concept in the GeoRaster data model. Layers are mapped to bands. Typically, one layer corresponds to one band. Layers are numbered from 1 to \( n \); that is, \( \text{layerNumber} = \text{bandNumber} + 1 \).

A **GeoRaster object** can contain multiple bands, which can also be called multiple layers.

**Interleaving:** Must be one of the following values: BSQ (band sequential), BIL (band interleaved by line), or BIP (band interleaved by pixel). Example: interleaving=BSQ
This figure shows an image with multiple layers and a single raster data table. Each layer contains multiple blocks, each of which typically contains many cells. Each block has an entry in the raster data table. Note that GeoRaster starts layer numbering at 1 and band numbering at 0 (zero).
Interleaving: Must be one of the following values: **BSQ (band sequential)**, BIL (band interleaved by line), or BIP (band interleaved by pixel). Example: interleaving=BSQ

Source: Esri ArcGIS’s home page
GeoRaster – Interleaving

**Interleaving**: Must be one of the following values: BSQ (band sequential), **BIL (band interleaved by line)**, or BIP (band interleaved by pixel). Example: interleaving=BSQ
GeoRaster – Interleaving

**Interleaving**: Must be one of the following values: BSQ (band sequential), BIL (band interleaved by line), or **BIP (band interleaved by pixel)**. Example: interleaving=BSQ

Source: Esri ArcGIS’s home page
IMPORTANT NOTES:

(1) SDO_GEOG.importFrom: This procedure does not support source multiband raster data with BIL and BSQ interleaving types. Only BIP interleaving.
Case Study

Region of interest: Novo Progresso, Pará

Images: 12 CBERS-2B scenes (CCD sensor)
Case Study - Images

Spatial reference system: UTM / WGS-84 Datum.
Spatial resolution: 20 meters
Radiometric resolution: 8 bits unsigned. It means that each image element, or pixel, has an integer value in the range of 0 to 255. The value 0 indicates pixels with “no data”, or with no valid information.
Each scene has 3 bands (2, 3 and 4)
Each scene is a GeoTIFF file.
Total: 36 files.
CREATE TABLE para_georaster
(
    r_georid NUMBER,
    r_scene VARCHAR(10),
    r_band NUMBER,
    r_satellite VARCHAR(20),
    r_date VARCHAR(20),
    r_image MDSYS.SDO_GEORASTER);

CREATE TABLE para_raster OF MDSYS.SDO_RASTER
(PRIMARY KEY (rasterID, pyramidLevel,
              bandBlockNumber, rowBlockNumber,
              columnBlockNumber));
Case Study – Import GeoTIFF files

One scene with its three bands is represented as a georaster object and stored in a row of the para_georaster table.

Tiles: 512 x 512 pixels.
Case Study – Import GeoTIFF files

We have 2 strategies to insert the GeoTIFF files (each file is one band) into the database using the package `sdo_geor`:

1. Import each file as it is, using the `sdo_geor.importFrom` function and, afterwards, use the `sdo_geor.mergeLayers` function to merge all bands of the same image into a single georaster object.

2. Create a new GeoTiff file combining all bands of a scene and, afterwards, use the `sdo_geor.importFrom` function to import the new file to a georaster object. The function `sdo_geor.importFrom` supports only multiple BIP (band interleaved by pixel) GeoTiff files.
Case Study – Import GeoTIFF files

DECLARE
  geor10 SDO_GEORASTER;
BEGIN
  INSERT INTO para_georaster VALUES(10, '167_108', 234, 'CBERS2B_CCD1XS', '20090820', sdo_geor.init('para_raster') );

  SELECT r_image INTO geor10 FROM para_georaster WHERE r_georid = 10 FOR UPDATE;

  sdo_geor.importFrom(geor10,'blocking=TRUE blocksize=(512,512) spatialExtent=TRUE srid=32721', 'GeoTIFF', 'file', '/home/..../file1.tif');

  UPDATE para_georaster SET r_image = geor10 WHERE r_georid = 10;
COMMIT;
END;

Using strategy 2
Case Study – Import GeoTIFF files

`sdo_geor.init` function: register automatically the new raster objects and their related raster data tables in the two metadata tables `user_sdo_geor_sysdata` and `all_sdo_geor_sysdata`.

`sdo_geor.importFrom` function: storage parameters (block size, compression type, pyramid generation using different resampling methods, …).

Important Note: to use the function `mdsys.sdo_geor.init`, your Oracle user must have permission to insert and update tables in the MDSYS schema!!!! Because of the metadata tables!
Case Study – Image Access

Return the raster value in a position given by a spatial coordinate (672512.103, 9214134.635):

```
SELECT sdo_geor.getCellValue(r_image, 0,
   sdo_geometry(2001, 32721,
   sdo_point_type(672512.103, 9214134.635, null),
   null, null), 1)
FROM para_georaster
WHERE r_georid = 10;
```
CASE STUDY – IMAGE ACCESS

Extract statistics and histogram:

DECLARE
    geor10  SDO_GEORASTER;
    window SDO_NUMBER_ARRAY := NULL;
BEGIN
    SELECT r_image INTO geor10 FROM para_georaster
    WHERE r_georid = 10 FOR UPDATE;

    sdo_geor.setBinFunction(geor10, 1,
                            sdo_number_array(0,10,1,0,255));

    sdo_geor.generateStatistics( geor10, 'samplingFactor=1',
                                  window,'TRUE','1','TRUE');

    UPDATE para_georaster SET r_image=geor10 WHERE r_georid=10;
END;
Case Study – Image Access

`sdo_geor.generateStatistics` function: extracts summarizing values, such as minimum, maximum and mean values. It can extract a histogram parameterized with a function to control the number of bins. It can also retrieve the statistics of a specific window within the image and to disregard `nodata` values.

This function stores the results in the georaster object metadata.
Case Study – Image Access

Get the store statistics and histogram from the georaster objects:

```
SELECT sdo_geor.getStatistics(r_image, 1)
FROM para_georaster WHERE r_georid = 10;

SELECT sdo_geor.getHistogram(r_image, 1)
FROM para_georaster WHERE r_georid = 10;
```
Case Study – Visualization

GeoRaster Viewer:
Georaster objects stores metadata about their visualization:

(1) Color map or *pallets*: mechanism to transform a range of input values into a range of colors (functions

\[
\text{sdo\_geor.setColorMapTable and sdo\_geor.setColorMap}
\]
Case Study – Visualization

Georaster objects stores metadata about their visualization:

(2) Association of image bands to Red-Green-Blue components of a display using the functions:

DECLARE
    geor SDO_GEORASTER;
BEGIN
    SELECT r_image INTO geor FROM para_georaster
    WHERE r_georid = 86 FOR UPDATE;

    sdo_geor.setDefaultRed(geor, 1);
    sdo_geor.setDefaultGreen(geor, 3);
    sdo_geor.setDefaultBlue(geor, 2);

    UPDATE para_georaster SET r_image = geor
    WHERE r_georid = 86;
COMMIT; END;
Case Study – Clip

We can clip a georaster object using a geometry (SDO_GEOMETRY) as a mask through the function `sdo_geor.subset`: