日日是Oracle APEX

Oracle APEXを使った作業をしていて、気の付いたところを忘れないようにメモをとります。

2021年7月6日火曜日

行政区域データのまとめ - 追記

こちらの記事で行政区域データを、行政区域コードや都道府県でまとめる作業を行いました。以下に、追加で確認したことを書き留めておきます。

SDO_UTIL.RECTIFY_GEOMETRYの結果

GeoJSONのデータを表JAR_JAPAN_REGION_ADMINSの列GEOMETRYに取り込んだ後に、SDO_UTIL.RECTIFY_GEOMETRYを実行しています。修正したジオメトリは列GEOM_Rへアップデートしています。

列GEOM_Rを確認すると、ジオメトリのSDO_GTYPEに2003(ポリゴン)だけではなく、2007(マルチポリゴン)も含まれています。確認に使用したコードは以下です。

```
set serveroutput on
declare
   l_geom sdo_geometry;
   l_elem_cnt number;
begin
   for c in (
      select * from jar_japan_region_admins
      where geom_r is not null order by id
   )
   loop
      l_geom := c.geom_r;
      l_elem_cnt := sdo_util.getnumelem(l_geom);
      if l_geom.sdo_gtype <> 2003 then
         dbms_output.put_line('id ' || c.id || ' is ' || l_geom.sdo_gtype || ', contains '
            || l_elem_cnt || ' elements.');
      end if;
   end loop;
end;
/
```

実行結果は以下のようになりました。

```
id 922 is 2007, contains 2 elements.
id 67318 is 2007, contains 2 elements.
id 78966 is 2007, contains 2 elements.
id 83266 is 2007, contains 2 elements.
id 94185 is 2007, contains 2 elements.
id 99633 is 2007, contains 2 elements.
id 100483 is 2007, contains 2 elements.
id 100866 is 2007, contains 2 elements.
id 101181 is 2007, contains 2 elements.
```

このままSDO_UTIL.SIMPLIFYを実行すると、含まれているポリゴンが非常に小さい場合、そのポリゴンが点または線のジオメトリに変換されます。結果としてOracle APEXのマップ・リージョンで表示されないジオメトリになります。

マルチポリゴンのジオメトリを分解し、別のIDを割り振って保存します。実行には以下のコードを使いました。

```
set serveroutput on
declare
   l_geom sdo_geometry;
   l_elem_cnt number;
   l_geom_in sdo_geometry;
   l_max number;
begin
   for c in (
      select * from jar_japan_region_admins
      where geom_r is not null order by id
   )
   loop
      -- rectifyされたジオメトリが処理の対象。
      l_geom := c.geom_r;
      --
      if l_geom.sdo_gtype in (2004,2007) then
         -- コレクション(2004)とマルチポリゴンが(2007)が修正の対象。
         dbms_output.put_line('id ' || c.id
            || ' was converted into collection(2004) or multipolygon(2007). try to fix.');
         l_elem_cnt := sdo_util.getnumelem(l_geom);
         select max(id) into l_max from jar_japan_region_admins;
         for i in 1..l_elem_cnt
         loop
            l_geom_in := sdo_util.extract(l_geom,i);
            if l_geom_in.sdo_gtype = 2003 then
               -- 含まれているポリゴンを新たなIDで追加する。
               dbms_output.put_line('store inner polygon with new id ' || (l_max+i));
               insert into jar_japan_region_admins(
                  id, prefecture_name, branch_in_hokkaido, major_city,
                  city_name, admin_code, geom_r, vertices)
               values(
                  l_max+i, c.prefecture_name, c.branch_in_hokkaido, c.major_city,
                  c.city_name, c.admin_code, l_geom_in, sdo_util.getnumvertices(l_geom_in));
            else
               -- それ以外は削除される。
               dbms_output.put_line('ignore inner geometry. ' || l_geom_in.sdo_gtype);
            end if;
         end loop;
         -- 元々のジオメトリは重複したデータになるためNULLにする。
         dbms_output.put_line('nullify original rectified geometry.');
         update jar_japan_region_admins set geom_r = null, vertices = null  where id = c.id;
      end if;
   end loop;
end;
/
commit;
```

実行結果は以下のようになりました。

```
id 922 was converted into collection(2004) or multipolygon(2007). try to
fix.
store inner polygon with new id 121159
store inner polygon with new id 121160
nullify original rectified geometry.
```

```
id 67318 was converted into collection(2004) or multipolygon(2007). try
to fix.
store inner polygon with new id 121161
store inner polygon with new id 121162
nullify original rectified geometry.
id 78966 was converted into collection(2004) or multipolygon(2007). try
to fix.
store inner polygon with new id 121163
store inner polygon with new id 121164
nullify original rectified geometry.
id 83266 was converted into collection(2004) or multipolygon(2007). try
to fix.
store inner polygon with new id 121165
store inner polygon with new id 121166
nullify original rectified geometry.
id 94185 was converted into collection(2004) or multipolygon(2007). try
to fix.
store inner polygon with new id 121167
store inner polygon with new id 121168
nullify original rectified geometry.
id 99633 was converted into collection(2004) or multipolygon(2007). try
to fix.
store inner polygon with new id 121169
store inner polygon with new id 121170
nullify original rectified geometry.
id 100483 was converted into collection(2004) or multipolygon(2007). try
to fix.
store inner polygon with new id 121171
store inner polygon with new id 121172
nullify original rectified geometry.
id 100866 was converted into collection(2004) or multipolygon(2007). try
to fix.
store inner polygon with new id 121173
store inner polygon with new id 121174
nullify original rectified geometry.
id 101181 was converted into collection(2004) or multipolygon(2007). try
to fix.
store inner polygon with new id 121175
store inner polygon with new id 121176
nullify original rectified geometry.
```

含まれているジオメトリはすべて、表JAR_JAPAN_ADMIN_REGIONSに追記できました。

SDO_UTIL.SIMPLIFYの結果

SDO_UTIL.RECTIFY_GEOMETRYの結果を対象として、SDO_UTIL.SIMPLIFYを実行します。閾値が10m、許容差として0.05(5cm)を指定しています。

```
update jar_japan_region_admins set geom_s = sdo_util.simplify(geom_r,10,0.05);
update jar_japan_region_admins set vert_s = sdo_util.getnumvertices(geom_s);
commit;
```

列GEOM_Rに含まれているポリゴンが小さいと、結果として点(SDO_GTYPEが2001)、または線(SDO_GTYPEが2002)になります。この時点でポリゴンではなくなるため、これ以降の処理の対象にする必要がありません。また、SDO_UTIL.SIMPLIFYでもコレクション(2004)やマルチポリゴン(2007)となる場合があります。

以下のコードでジオメトリの確認を行います。

```
set serveroutput on
declare
   l_geom sdo_geometry;
   l_elem_cnt number;
   l_point_cnt integer := 0;
   l_line_cnt  integer := 0;
begin
   for c in (
      select * from jar_japan_region_admins
      where geom_s is not null
      order by id
   )
   loop
      l_geom := c.geom_s;
      l_elem_cnt := sdo_util.getnumelem(l_geom);
      if l_geom.sdo_gtype = 2001 then
         l_point_cnt := l_point_cnt + 1;
      elsif l_geom.sdo_gtype = 2002 then
         l_line_cnt := l_line_cnt + 1;
      elsif l_geom.sdo_gtype = 2003 then
         -- polygon is valid
         continue;
      else
         dbms_output.put_line('id ' || c.id || ' is ' || l_geom.sdo_gtype || '. contains '
            || l_elem_cnt || ' elements.');
      end if;
   end loop;
   dbms_output.put_line('number of point exists. ' || l_point_cnt);
   dbms_output.put_line('number of line exists. '  || l_line_cnt);
end;
/
```

実行結果は以下のようになりました。

```
id 658 is 2007. contains 2 elements.
id 789 is 2007. contains 2 elements.
id 1286 is 2007. contains 4 elements.
[中略]
id 117307 is 2007. contains 3 elements.
id 119059 is 2004. contains 2 elements.
id 121174 is 2007. contains 2 elements.
number of point exists. 50702
number of line exists. 40409
```

多くのデータがコレクションまたはマルチポリゴンに変換されています。また、点や線になったポリゴンも多数あります。

列GEOM_Sで点や線になっているジオメトリをNULLに置き換えます。また、コレクションやマルチポリゴンは分解して、新たなIDで表JAR_JAPAN_REGION_ADMINSに保存します。実行のために以下のコードを使用しました。

```
set serveroutput on
declare
   l_geom sdo_geometry;
   l_elem_cnt number;
   l_geom_in sdo_geometry;
   l_max number;
begin
```

```
  for c in (
    select * from jar_japan_region_admins
    where geom_s is not null order by id
  )
  loop
    -- simplifyされたジオメトリが処理の対象。
    l_geom := c.geom_s;
    --
    if l_geom.sdo_gtype in (2001,2002) then
      /* 大量にあるので表示しない。
      dbms_output.put_line('id ' || c.id
        || ' is converted into point(2001) or line(2002). nullfiy.');
      */
      update jar_japan_region_admins set geom_s = null, vert_s = null where id = c.id;
    elsif l_geom.sdo_gtype in (2004,2007) then
      -- コレクション(2004)とマルチポリゴンが(2007)が修正の対象。
      dbms_output.put_line('id ' || c.id
        || ' was converted into collection(2004) or multipolygon(2007). try to fix.');
      l_elem_cnt := sdo_util.getnumelem(l_geom);
      select max(id) into l_max from jar_japan_region_admins;
      for i in 1..l_elem_cnt
      loop
        l_geom_in := sdo_util.extract(l_geom,i);
        if l_geom_in.sdo_gtype in (2001,2002) then
          -- 点と線は削除する。
          dbms_output.put_line('ignore inner geometry. ' || l_geom_in.sdo_gtype);
        else
          -- それ以外は追加する。
          dbms_output.put_line('store inner polygon with new id ' || (l_max+i));
          insert into jar_japan_region_admins(
            id, prefecture_name, branch_in_hokkaido, major_city,
            city_name, admin_code, geom_s, vert_s)
          values(
            l_max+i, c.prefecture_name, c.branch_in_hokkaido, c.major_city,
            c.city_name, c.admin_code, l_geom_in, sdo_util.getnumvertices(l_geom_in));
        end if;
      end loop;
      -- 元々のジオメトリはNULLにする。
      dbms_output.put_line('nullify original simplified geometry.');
      update jar_japan_region_admins set geom_s = null, vert_s = null where id = c.id;
    end if;
  end loop;
end;
/
commit;
```

再度、列GEOM_Sのチェックスクリプトを実行すると以下の結果になりました。

```
number of point exists. 0
number of line exists. 0
```

列GEOM_Sには、１行につき１つの単純なポリゴンが含まれています。

行政区域コード単位での集計

列GEOM_Sに含まれていた集計対象にならないジオメトリは、すでにNULLになっています。そのため面積を計算して、集計対象を制限する必要はありません。列GEOM_SがNULLの行を集計対象から

外します。

MERGE文は以下になります。

```
merge into jar_japan_region_cities c
using
(
   select
      prefecture_name, branch_in_hokkaido, major_city, city_name
      ,admin_code, geom_s geometry, vert_s vertices
   from jar_japan_region_admins
   where
     geom_s is not null
     and
     (prefecture_name, admin_code)
     in
     (
        select prefecture_name, admin_code
        from jar_japan_region_admins
        group by prefecture_name, admin_code
        having count(*) = 1
     )
) a
on (c.prefecture_name = a.prefecture_name and c.admin_code = a.admin_code)
when matched then
   update set
      branch_in_hokkaido = a.branch_in_hokkaido
      ,major_city = a.major_city
      ,city_name = a.city_name
      ,geometry = a.geometry
      ,row_count = 1
      ,vertices = a.vertices
when not matched then
   insert (prefecture_name, branch_in_hokkaido, major_city, city_name
      ,admin_code, geometry, row_count, vertices)
   values (a.prefecture_name, a.branch_in_hokkaido, a.major_city, a.city_name
      ,a.admin_code, a.geometry, 1, a.vertices)
;
```

２つ以上のデータがある行政区域コードをまとめるために使用するコードは以下になります。

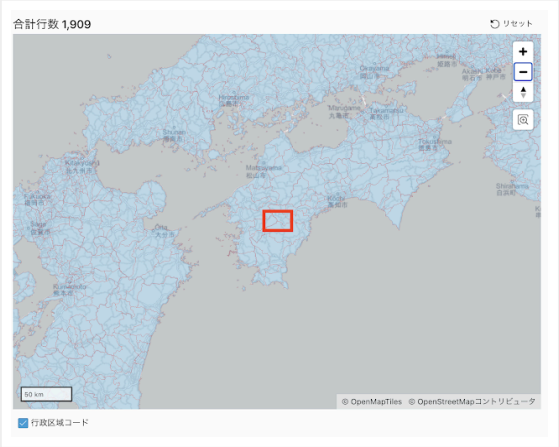
```
declare
   l_geom   sdo_geometry;
   l_geom_a sdo_geometry;
   l_row_count number;
   l_vertices number;
   l_message varchar2(4000);
begin
   delete from jar_validate_cities;
   commit;
   for c in (
      -- まとめる数が少ない行政区分からひとつひとつまとめていく。
      select prefecture_name, branch_in_hokkaido, major_city, city_name, admin_code
      from jar_japan_region_admins
      where
      geom_s is not null
      and
      (prefecture_name, admin_code) not in (
          -- すでにまとめ済みの行政区分は除外する
          select prefecture_name, admin_code from jar_japan_region_cities
      )
      group by prefecture_name, branch_in_hokkaido, major_city, city_name, admin_code
```

```
      order by count(*) asc
    )
    loop
      l_geom := null;
      for r in
      (
        select prefecture_name, admin_code, geom_s geometry, vert_s vertices
        from jar_japan_region_admins
        where prefecture_name = c.prefecture_name
          and admin_code = c.admin_code
          and geom_s is not null
      )
      loop
        if l_geom is null then
          -- 最初の1行は初期化に使用する。
          l_geom := r.geometry;
          l_row_count := 1;
          l_vertices := r.vertices;
          continue;
        end if;
        -- ポリゴン、行数、頂点の数を集計する
        if sdo_geom.relate(l_geom, 'disjoint', r.geometry, 0.05) = 'TRUE' then
          l_geom_a := sdo_util.append(l_geom, r.geometry);
        else
          l_geom_a := sdo_geom.sdo_union(l_geom, r.geometry, 0.05);
        end if;
        l_geom := l_geom_a;
        l_row_count := l_row_count + 1;
        l_vertices := l_vertices + r.vertices;
      end loop;
      -- 結果を保存する。途中で停止できるよう毎回commitする。
      l_message := sdo_geom.validate_geometry_with_context(l_geom, 0.05);
      insert into jar_validate_cities(prefecture_name, branch_in_hokkaido,
        major_city, city_name, admin_code, message)
      values(
        c.prefecture_name, c.branch_in_hokkaido,
        c.major_city, c.city_name, c.admin_code, l_message);
      insert into jar_japan_region_cities
        (prefecture_name, branch_in_hokkaido, major_city, city_name
        ,admin_code, geometry, row_count, vertices)
      values(
        c.prefecture_name, c.branch_in_hokkaido, c.major_city, c.city_name,
        c.admin_code, l_geom, l_row_count, l_vertices);
      commit;
    end loop;
end;
/
```

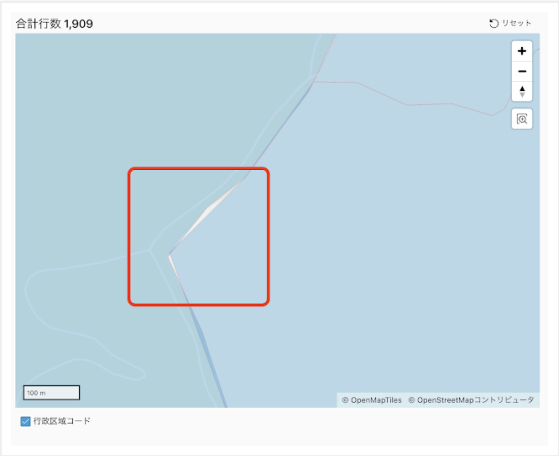
集計の対象外になったデータは、SDO_UTIL.SIMPLIFYを使って閾値10m、許容差0.05(5cm)で簡略化した結果がポリゴンにならなかったもの、ということになります。

行政区域の隙間

以下の倍率程度であれば、行政区域が接する部分に隙間は見えません。



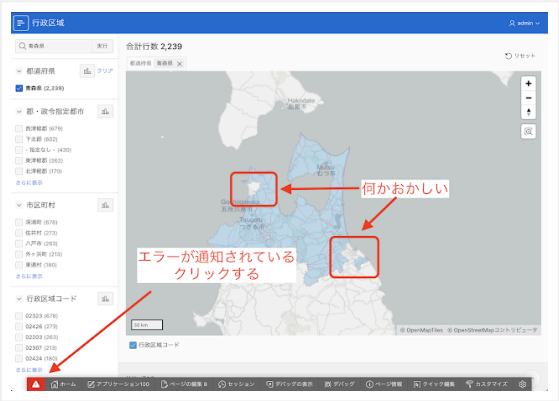
拡大すると、行政区域の接する部分に隙間があることがわかります。



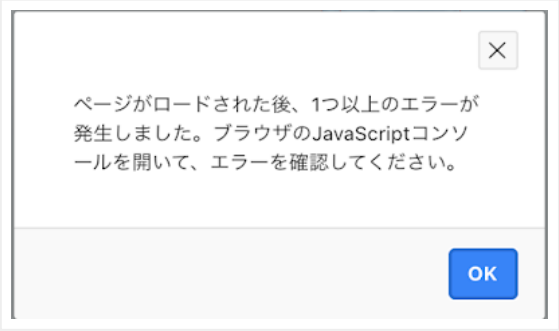
単純にSDO_UTIL.SIMPLIFYを使用して頂点を間引く方法では、このような隙間が発生することは避けられないです。

ジオメトリのエラーを見つける

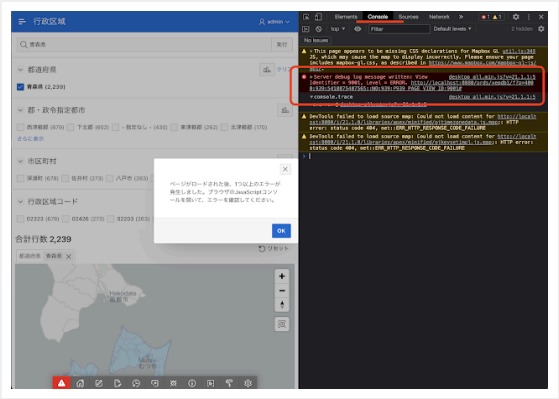
ジオメトリに限りませんが、エラーが発生していると開発者ツール・バーに通知されます。



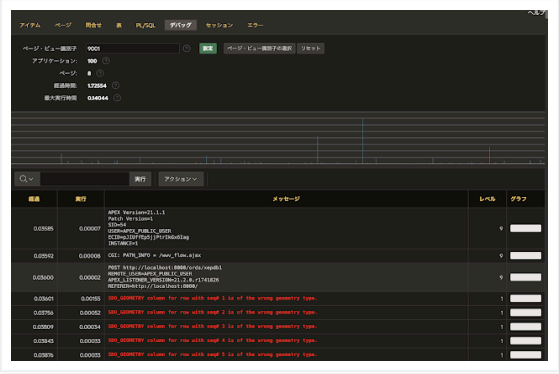
エラーのアイコンをクリックすると、ブラウザのJavaScriptコンソールを開いて、エラーを確認してください。とあります。



開いてみます。Consoleタブを見ると、Server debug log message written: View Identifier = 9001, Level = ERROR. とあり、リンクが続いています。リンクをクリックします。



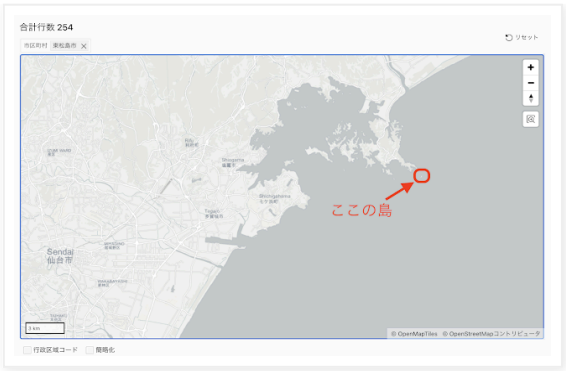
エラー・メッセージを確認することができます。今回のエラーはSDO_GEOMETRY column for row id with seq# 1 is of the wrong geometry type. ということで、ジオメトリのタイプが適切でないためにエラーが発生していることがわかります。



ブラウザのJavaScirptコンソールを開いたり、ひと手間ありますが、デバッグの役に立つ機能です。

簡略化されたデータの比較

宮城県、東松島市の波島を見てみます。



OpenStreetMapのバックグランドは以下です。

Oracle World Mapのバックグランドは以下です。島がないです。

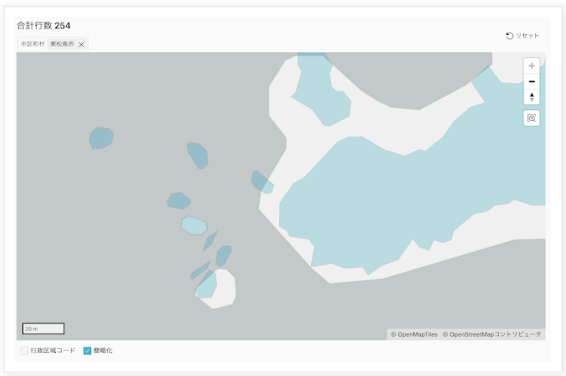
表示している場所に間違いはありません。

SDO_UTIL.RECTIFY_GEOMETRYを、許容差0.05で実行した結果です。

SDO_UTIL.SIMPLIFYを、閾値1m、許容差0.05で実行した結果です。

もっと拡大して確認してみます。簡略化前です。

簡略化後です。それほど違いが見られません。



頂点の数は1/2程度になっています。それほど精度は落ちていません。

```
SQL> select sum(vertices), sum(vert_s) from jar_japan_region_admins;
```

```
SUM(VERTICES) SUM(VERT_S)
------------- -----------
     14748036     6734337
```

SDO_UTIL.SIMPLIFYを、閾値10m、許容差0.05で実行した結果です。小さな島が無くなり、簡略化されていることが見て分かります。

頂点の数は1/10程度になっています。

```
SQL> select sum(vertices), sum(vert_s) from jar_japan_region_admins;

SUM(VERTICES) SUM(VERT_S)
------------- -----------
     14748036     1647760
```

完

Yuji N. 時刻: 14:52

共有

‹ ホーム ›

ウェブ バージョンを表示

自己紹介

Yuji N.

日本オラクル株式会社に勤務していて、Oracle APEXのGroundbreaker Advocateを拝命しました。
こちらの記事につきましては、免責事項の参照をお願いいたします。

詳細プロフィールを表示