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9日金曜日

空間索引の作成と利用について

こちらの記事で作成したアプリケーションで、Oracle APEXの**自動化**を使い、RESTサービスから取得できる医療提供体制の状況をローカル表に定期的に同期させようとしています。その際に、居住地近辺の医療機関を対象とするためにOracle Spatialの機能を使うことにしました。

以下、同期化の対象を選ぶために実施した作業のログです。Oracle Spatialの空間索引を使い、効果を確認しています。

地域を限定する

オープンデータは全国を網羅しています。身近な用途でデータを活用することを想定し、同期するデータを居住地の近隣に限定します。

以前に国土交通省が提供している行政区域データを取り込む方法について、こちらの記事で紹介しています。この記事で作成した表**JAR_JAPAN_REGION_CITIES**を使って、居住地の座標から行政区域を確認します。

居住地の緯度経度を139.625868,35.607440として、以下のSELECT文を実行します。

```
select n.prefecture_name, n.major_city, n.city_name, n.admin_code
from jar_japan_region_cities n
where sdo_anyinteract(n.geometry,
    sdo_geometry(2001, 4326,
        sdo_point_type(139.625868,35.607440,null),
        null, null
    )
) = 'TRUE';
```

以下の結果が得られます。神奈川県川崎市高津区、行政区域のコードは14134です。10数秒処理に時間がかかります。

PREFECTURE_NAME	MAJOR_CITY	CITY_NAME	ADMIN_CODE
神奈川県	川崎市	高津区	14134

高津区に隣接する行政区域を調べます。以下のSELECT文を実行します。

```
select n.admin_code, n.prefecture_name, n.major_city, n.city_name
from
(select * from jar_japan_region_cities) n,
(select admin_code, geometry from jar_japan_region_cities where admin_code = '14134') m
where sdo_anyinteract(n.geometry, m.geometry) = 'TRUE'
```

高津区を含め、以下の7つの行政区域が返されます。処理に数分かかる場合もあります。

ADMIN_CODE	PREFECTURE_NAME	MAJOR_CITY	CITY_NAME
14109	神奈川県	横浜市	港北区
14135	神奈川県	川崎市	多摩区

14134	神奈川県	川崎市	高津区
13112	東京都	-	世田谷区
14133	神奈川県	川崎市	中原区
14136	神奈川県	川崎市	宮前区
14118	神奈川県	横浜市	都筑区

この７つの行政区域を対象にしてデータの同期をしたかったのですが、オープンデータに含まれている自治体コードはもっと大きな単位でした。そのため、データの取得条件として有効な自治体コードを探してみます。

以下のSQLを実行します。上記の７つの自治体に含まれている医療機関を検索し、その医療機関に割当たっている自治体コードを調べます。

```
select f.local_gov_code from
c19_medical_facility_statuses f,
(
    select geometry from jar_japan_region_cities
    where admin_code in ('14109','14135','14134','13112','14133','14136','14118')
) r
where sdo_anyinteract(r.geometry,
    sdo_geometry(2001,4326,
        sdo_point_type(f.longitude,f.latitude,null),
        null, null
    )
) = 'TRUE'
group by f.local_gov_code;
```

以下の結果を得ることができました。結果を得るまでに10分弱の時間がかかっています。

LOCAL_GOV_CODE
141003
141305
131121

最初に作成した表**MUNICIPALITY_CODES**から**団体コード**を検索してみます。

```
select * from municipality_codes where "団体コード" in (141003,141305,131121)
```

以下の結果が得られます。

ID	団体コード	都道府県名__漢字_	市区町村名__漢字_	都道府県名__ｶﾅ_	市区町村名__ｶﾅ_
658	131121	東京都	世田谷区	ﾄｳｷｮｳﾄ	ｾﾀｶﾞﾔｸ
710	141003	神奈川県	横浜市	ｶﾅｶﾞﾜｹﾝ	ﾖｺﾊﾏｼ
711	141305	神奈川県	川崎市	ｶﾅｶﾞﾜｹﾝ	ｶﾜｻｷｼ

東京都世田谷区 - **131121**、神奈川県横浜市 - **141003**、神奈川県川崎市 - **141305**を対象として、データの同期を行うことにします。

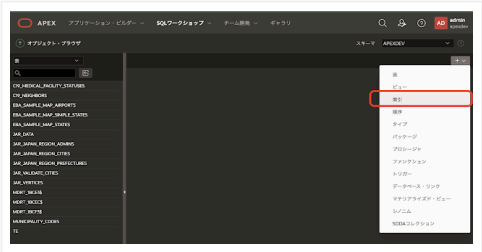
空間索引を作成する

今まで実行したSELECT文では明らかに、Oracle SpatialのSDO_ANYINTERACT空間演算子の処理に時間がかかっています。

Oracle APEX 21.1より空間索引を作成するウィザードがオブジェクト・ブラウザに追加されています。この機能を使用して空間索引を作成することにより、処理速度を上げてみます。

表JAR_JAPAN_REGION_CITIESの**列GEOMETRY**に空間索引を作成します。

SQLワークショップより**オブジェクト・ブラウザ**を開き、**作成**メニューから**索引**を実行します。



表名として**JAR_JAPAN_REGION_CITIES**を選択し、**索引のタイプ**を**空間**とします。**次**へ進みます。

索引名は自動的に**JAR_JAPAN_REGION_CITIES_SX**になります。**索引列**として**GEOMETRY**を選択します。**システム管理索引**は**ON**が推奨です。変更はしません。この表の列GEOMETRYに含まれるジオメトリはすべてポリゴンなので、**表にはポイント・オブジェクトのみが含まれる**は**OFF**です。列の登録へ進みます。

空間メタデータの追加画面になります。Oracle SpatialでのUSER_SDO_GEOM_METADATAビューの更新処理にあたります。**座標系**は索引を作成する列に含まれているジオメトリのSDO_SRIDより決まり、**空間許容範囲**(Oracle Spatialのドキュメントでは**許容差**と訳されています)は、この列を扱う操作で指定する予定の値になります。

空間メタデータの追加をクリックします。

続いて表示される確認画面にて、実行されるSQLを確認できます。Oracle APEXでは空間メタデータの追加と削除を行うプロシージャーを提供しています。ここでは**APEX_SPATILAL.INSERT_GEOM_METADATA**を呼び出しています。**実行**をクリックします。

空間メタデータの追加が終了し、空間索引の作成画面に戻ります。**次**へ進みます。

確認画面が表示されます。必須の作業ではありませんが、**SQL**をクリックして実行されるSQLを確認してみましょう。確認したら、**索引の作成**をクリックします。

以上で表JAR_JAPAN_REGION_CITIESの列GEOMETRYに空間索引**JAR_JAPAN_REGION_CITIES_SX**が作成されました。

空間索引の効果を確認する

先程実行したSELECT文を、再度実行してみます。

```
select n.prefecture_name, n.major_city, n.city_name, n.admin_code
from jar_japan_region_cities n
where sdo_anyinteract(n.geometry,
    sdo_geometry(2001, 4326,
        sdo_point_type(139.625868,35.607440,null),
        null, null
    )
) = 'TRUE';
```

空間索引作成前は３度実行して、13.50秒, 14.36秒, 13.79秒でした。空間索引作成後は、0.20秒、0.04秒、0.01秒になっています。初回に比較的時間がかかるのは、索引をバッファキャッシュに乗せているためだと思われます。２回目以降はさらに速くなっています。

```
select n.admin_code, n.prefecture_name, n.major_city, n.city_name
from
(select * from jar_japan_region_cities) n,
(select admin_code, geometry from jar_japan_region_cities where admin_code = '14134') m
where sdo_anyinteract(n.geometry, m.geometry) = 'TRUE'
```

空間索引作成前は３度実行して、14.95秒, 15.75秒, 15.20秒でした。空間索引作成後は、0.19秒、0.05秒、0.05秒になっています。

注意が必要な点ですが、SDO_ANYINTERACT演算子では、空間索引が使用されるのは最初の引数にたいしてです。SDO_ANYINTERACT演算子であれば、第１引数と第２引数を入れ替えても結果は同じですが、処理速度は変わります。

例えば、1秒以下で終了する２番目のSELECT文のsdo_anyinteract(n.geometry,m.geometry)の引数の順序を変えてsdo_anyinteract(m.geometry,n.geometry)として実行してみます。

```
select n.admin_code, n.prefecture_name, n.major_city, n.city_name
from
(select * from jar_japan_region_cities) n,
(select admin_code, geometry from jar_japan_region_cities where admin_code = '14134') m
where sdo_anyinteract(m.geometry, n.geometry) = 'TRUE'
```

結果が返ってくるまでに103.96秒かかりました。第１引数のm.geometryはadmin_code = '14134'の条件があるため1行です。それに対してn.geometryは1909行ですが、２つ目の引数なので空間索引は使用されません。そのため速度に違いが生じます。

続いて、以下のSQLの改善を検討します。

```
select f.local_gov_code from
c19_medical_facility_statuses f,
(
    select geometry from jar_japan_region_cities
    where admin_code in ('14109','14135','14134','13112','14133','14136','14118')
) r
where sdo_anyinteract(r.geometry,
    sdo_geometry(2001,4326,
        sdo_point_type(f.longitude,f.latitude,null),
        null, null
    )
) = 'TRUE'
group by f.local_gov_code;
```

416.70秒の処理時間がかかっています。列LONGITUDE、LATITUDEにたいして空間索引を作成するため、表C19_MEDICAL_FACILILTY_STATUSESにSDO_GEOMETRY型の列**GEOMETRY**を追加します。

```
alter table c19_medical_facility_statuses add (geometry sdo_geometry);
```

続いて、列LONGITUDE, LATITUDEをSDO_GEOMETRY型のGEOMETRYに更新します。

```
update c19_medical_facility_statuses
set geometry = sdo_geometry(2001,4326,
        sdo_point_type(longitude,latitude,null),
        null,null);
```

作成した列GEOMETRYに空間索引を作成します。先程の空間索引JAR_JAPAN_REGION_CITIES_SXの作成手順と同様なので、スクリーンショットは省略します。ただし、**表にはポイント・オブジェクトのみが含まれる**はONにします。

空間索引が作成されたら、変更した以下のSELECT文を実行します。sdo_anyinteractの第１引数は医療機関の位置で２万行以上あります。対して行政区域は７行のみです。

```
select f.local_gov_code from
c19_medical_facility_statuses f,
(
    select geometry from jar_japan_region_cities
    where admin_code in ('14109','14135','14134','13112','14133','14136','14118')
) r
where sdo_anyinteract(f.geometry, r.geometry) = 'TRUE'
```

```
group by f.local_gov_code;
```

３回連続で実施して、0.13秒、0.01秒、0.01秒で結果が返ってきました。

この速度であれば、対話的な操作でも、Oracle Spatialの機能を使った問い合わせを実行できそうです。

空間メタデータを確認する

空間索引のメタデータは、Oracle APEXのウィザードが追加します。登録されている空間メタデータはビューUSER_SDO_GEOM_METADATAより確認できます。Oracle APEXの**SQLコマンド**からはSDO_DIM_ARRAY型の印刷はできないので、以下のPL/SQLプロシージャを使用して確認します。

```
declare
    l_dim_array sdo_dim_array;
    l_dim_info sdo_dim_element;
begin
    for r in (
        select * from user_sdo_geom_metadata
        where table_name in (
            'JAR_JAPAN_REGION_CITIES'
            ,'C19_MEDICAL_FACILITY_STATUSES'
        )
    )
    loop
        dbms_output.put_line('TABLE_NAME: ' || r.table_name || ' COLUMN_NAME: ' || r.column_name);
        l_dim_array := r.diminfo;
        for i in 1..2
        loop
            l_dim_info := l_dim_array(i);
            dbms_output.put_line(
                'DIMNAME: '    || l_dim_info.sdo_dimname ||
                ' LB: '        ||  l_dim_info.sdo_lb ||
                ' UB: '        || l_dim_info.sdo_ub ||
                ' TOLERANCE: ' || l_dim_info.sdo_tolerance
            );
        end loop;
    end loop;
end;
```

以下の結果が得られます。

```
TABLE_NAME: C19_MEDICAL_FACILITY_STATUSES COLUMN_NAME: GEOMETRY
DIMNAME: X LB: -180 UB: 180 TOLERANCE: .05
DIMNAME: Y LB: -90 UB: 90 TOLERANCE: .05
TABLE_NAME: JAR_JAPAN_REGION_CITIES COLUMN_NAME: GEOMETRY
DIMNAME: X LB: -180 UB: 180 TOLERANCE: .05
DIMNAME: Y LB: -90 UB: 90 TOLERANCE: .05
```

空間メタデータは索引を削除しても残ります。索引の許容差を変更する場合は空間メタデータも削除する必要があります。Oracle APEXではAPEX_SPATIALパッケージより、プロシージャDELETE_GEOM_METADATAを提供しています。空間索引の削除と同時に空間メタデータの削除を行うことができます。

例えば、以下のPL/SQLコードによって、今回作成した空間索引の削除ができます。

表JAR_JAPAN_REGION_CITIESの場合です。

```
begin
    apex_spatial.delete_geom_metadata(
            'JAR_JAPAN_REGION_CITIES','GEOMETRY',TRUE
    );
```

```
end;
```

表C19_MEDICAL_FACILITY_STATUSESの場合です。

```
begin
    apex_spatial.delete_geom_metadata(
            'C19_MEDICAL_FACILITY_STATUSES','GEOMETRY',TRUE
    );
end;
```

以上で空間索引の紹介は終了です。

Oracle APEXのアプリケーション作成の参考になれば幸いです。

完

Yuji N. 時刻: 23:46

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