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。

2020年7月20日月曜日

Oracle APEXでのデータ・ロード(4) - APEX_DATA_PARSERパッケージ

これまでの記事で、Oracle APEXに組み込まれているツールであるデータ・ワークショップ、および、Oracle APEXのアプリケーションに組み込まれるデータ・ロード・ウィザードについて説明しました。どちらのツールも、エンド・ユーザーが操作を行いデータのロードを行います。

今回の一連の記事は、新型コロナウイルス感染症の陽性患者属性をデータベースに取り込むことを題材にしています。陽性患者属性のデータは頻繁に更新されているため、最新の情報を取り込むためには、高い頻度でデータの再アップロードを行う必要があります。これを都道府県毎に行うのは大変な作業です。また、取り込むファイルのフォーマットは、基本的には都道府県単位では変わらないため、列のマッピングの指定をその都度行うのも非効率です。

Oracle APEXでは、PL/SQL APIとしてAPEX_DATA_PARSERパッケージを公開しています。マニュアルでの記載は以下です。日本語の記述はバージョンが古いので注意してください。

日本語マニュアルの記述(v19.1): https://docs.oracle.com/cd/F23071_01/aeapi/APEX_DATA_PARSER.html
英語マニュアルの説明(v20.1): https://docs.oracle.com/en/database/oracle/application-express/20.1/aeapi/APEX_DATA_PARSER.html

APEX_DATA_PARSERパッケージはデータ・ワークショップおよびデータ・ロード・ウィザードの内部で使用されているパッケージであり、これらのツールでロード可能なファイルは、すべてAPEX_DATA_PARSERパッケージで扱うことができます。

それでは、APEX_DATA_PARSERパッケージを使ってデータをロードするコードを書いてみます。

CSVファイルのロード

東京都が提供しているCSVファイルをAPEX_DATA_PARSERパッケージに含まれるPARSEファンクションを使ってパースし、表COVID19_PATIENTSに取り込むコードを書いてみます。

新型コロナウイルス陽性患者発表詳細として、東京都・オープンデータ・カタログサイトに掲載されています。
https://catalog.data.metro.tokyo.lg.jp/dataset/t000010d0000000068

CSVファイルは以下のURLで提供されています。
https://stopcovid19.metro.tokyo.lg.jp/data/130001_tokyo_covid19_patients.csv

Oracle APEXでは、APEX_WEB_SERVICEパッケージとして、アクセスしたURLのデータをCLOBまたはBLOBとして返すファンクションを提供しています。

APEX_WEB_SERVICE.MAKE_REST_REQUEST(CLOB)およびAPEX_WEB_SERVICE.MAKE_REST_REQUEST_B(BLOB)ファンクションですが、その名前のせいか、RESTサービスの呼び出しにしか使えないと思われてることがあります。そのようなことはなく、一般的なHTTPのリクエストの発行に使用することができます。

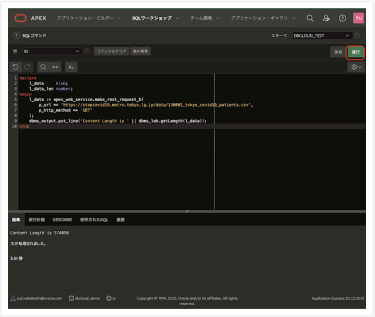
日本語マニュアルの記述(v19.1): https://docs.oracle.com/cd/F23071_01/aeapi/MAKE_REST_REQUEST-Function.html
https://docs.oracle.com/cd/F23071_01/aeapi/MAKE_REST_REQUEST_B-Function.html
英語マニュアルの説明(v20.1): https://docs.oracle.com/en/database/oracle/application-express/20.1/aeapi/MAKE_REST_REQUEST-Function.html
https://docs.oracle.com/en/database/oracle/application-express/20.1/aeapi/MAKE_REST_REQUEST_B-Function.html

最初にAPEX_WEB_SERVICEパッケージを使って、東京都が提供しているデータを取得できることを確認します。SQLワークショップのSQLコマンドから、PL/SQLスクリプトを実行します。

実行するPLSQLスクリプトは以下になります。

```
declare
   l_data     blob;
   l_data_len number;
begin
   l_data := apex_web_service.make_rest_request_b(
      p_url => 'https://stopcovid19.metro.tokyo.lg.jp/data/130001_tokyo_covid19_patients.csv',
      p_http_method => 'GET'
   );
   dbms_output.put_line('Content Length is ' || dbms_lob.getLength(l_data));
end;
```

以下の実行画面になります。



正常にファイルが取得されていれば、Content Length is ...と**結果**に表示されます。CSVファイルの取得は、APEX_WEB_SERVICE.MAKE_REST_REQUEST_Bの呼び出しで完了します。

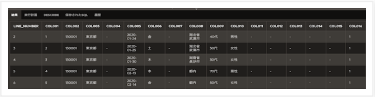
次に取得したCSVファイルのデータをパースします。APEX_DATA_PARSER.PARSEファンクションを使用します。APEX_DATA_PARSER.PARSEファンクションは表関数なので、受け取ったデータをパースして、結果を表形式で返します。

以下のSQLで取得したCSVファイルを表形式にパースします。一行のSELECT文で完了します。

```
select * from
   apex_data_parser.parse(
      p_content => apex_web_service.make_rest_request_b(
         p_url => 'https://stopcovid19.metro.tokyo.lg.jp/data/130001_tokyo_covid19_patients.csv',
         p_http_method => 'GET'
      ),
      p_file_name => 'file_is.csv',
      p_skip_rows => 1
   ) where col001 is not null
```

p_contentとして、取得したCSVの内容を渡しています。p_file_nameはfile_is.csvとしていますが、これはファイル名の拡張子よりファイルタイプを判別させるためで、ファイル名はそれ以外の用途では使われていません(p_file_typeという引数でCSVファイルとして2を指定する方法もあります)。最初の行は項目名なので、１行読み飛ばします。

SQLコマンドで実行すると結果は以下になります。



パースされた結果を使って、表COVID19_PATIENTSの内容を入れ替えるには、以下のSQLを実行します。DELETE/INSERTの2行です。

```
delete from covid19_patients where prefecture_name = '東京都';

insert into covid19_patients(
   "No",
   municipality_code,
   prefecture_name,
   published_date,
   patient_location,
   patient_age,
   patient_sex,
   patient_left_hospital
)
select
   to_number(col001),
   to_number(col002),
   col003,
   to_date(col005, 'YYYY-MM-DD'),
   col008,
   col009,
   col010,
   to_number(col016)
from
   apex_data_parser.parse(
      p_content => apex_web_service.make_rest_request_b(
         p_url => 'https://stopcovid19.metro.tokyo.lg.jp/data/130001_tokyo_covid19_patients.csv',
         p_http_method => 'GET'
      ),
      p_file_name => 'file_is.csv',
      p_skip_rows => 1
   ) where col001 is not null
```

完全に入れ替えるのではなく、差分を更新する場合は、MERGE文を使います。こちらは1行です。(とは言っても、長いですけど。。。)

```
merge into covid19_patients p
using
(
   select
      to_number(col001) "No",
      to_number(col002) municipality_code,
      col003 prefecture_name,
      to_date(col005, 'YYYY-MM-DD') published_date,
      col008 patient_location,
      col009 patient_age,
      col010 patient_sex,
```

```
      to_number(col016) patient_left_hospital
    from
      apex_data_parser.parse(
        p_content => apex_web_service.make_rest_request_b(
          p_url => 'https://stopcovid19.metro.tokyo.lg.jp/data/130001_tokyo_covid19_patients.csv',
          p_http_method => 'GET'
        ),
        p_file_name => 'file_is.csv',
        p_skip_rows => 1
      ) where col001 is not null
    minus
    select
      "No", municipality_code, prefecture_name, published_date,
      patient_location, patient_age, patient_sex, patient_left_hospital
    from covid19_patients
) n
on (p."No" = n."No" and p.prefecture_name = n.prefecture_name)
when matched then
    update set
      p.published_date   = n.published_date,
      p.patient_location = n.patient_location,
      p.patient_age      = n.patient_age,
      p.patient_sex      = n.patient_sex,
      p.patient_left_hospital = n.patient_left_hospital
when not matched then
    insert(
      "No", municipality_code, prefecture_name, published_date,
      patient_location, patient_age, patient_sex, patient_left_hospital
    )
    values
    (
      n."No", n.municipality_code, n.prefecture_name, n.published_date,
      n.patient_location, n.patient_age, n.patient_sex, n.patient_left_hospital
    );
```

INSERT文とMERGE文の両方で、以下のSELECT文が使われています。

```
select
    to_number(col001) "No",
    to_number(col002) municipality_code,
    col003 prefecture_name,
    to_date(col005, 'YYYY-MM-DD') published_date,
    col008 patient_location,
    col009 patient_age,
    col010 patient_sex,
    to_number(col016) patient_left_hospital
from
    apex_data_parser.parse(
      p_content => apex_web_service.make_rest_request_b(
        p_url => 'https://stopcovid19.metro.tokyo.lg.jp/data/130001_tokyo_covid19_patients.csv',
        p_http_method => 'GET'
      ),
      p_file_name => 'file_is.csv',
      p_skip_rows => 1
    ) where col001 is not null
```

都道府県より新型コロナウイルス感染症の陽性患者属性のデータがCSVやExcel形式で提供されている場合は、上記のSELECT文の記述を変更することで、データを標準化して表COVID19_PATIENTSへ投入することができます。

集合演算子のMINUSを使っている以下の部分は、SQLから削除しても結果は変わりません。

```
    minus
    select
      "No", municipality_code, prefecture_name, published_date,
      patient_location, patient_age, patient_sex, patient_left_hospital
    from covid19_patients
```

この部分で、変更のないデータは評価の対象から外しているため、アップデートから除外されます。MINUSをしていない場合は、同じデータによるアップデートが実行されます。

取得したデータのキャッシュ

今までの方法だと、SQL文を実行するたびにオープンデータへのアクセスが発生します。特に開発時にはSQLの実行を何回も行うことになるため、あまり行儀がよくありません。取得したデータは一旦表のBLOB列に保存して、SQLからはそちらを参照するようにします。

すでに作成済みの表COVID19_MUNICIPALITIES表にBLOBデータを保持する列(CONTENT_BLOB)と保存を実行した日時(LAST_UPDATE_DATE)、ファイル名(FILE_NAME)、取得したデータのURL(CONTENT_URL)を追加します。

```
alter table covid19_municipalities add content_blob blob;

alter table covid19_municipalities add last_update_date timestamp with local time zone;

alter table covid19_municipalities add file_name varchar2(200);

alter table covid19_municipalities add content_url varchar2(400);
```

これらを追加した上で、データの取得処理とパース処理を分けて実行します。

東京都のデータ取得処理は以下になります。

```
declare
  l_url covid19_municipalities.content_url%type;
  l_file_name covid19_municipalities.file_name%type;
begin
  l_url := 'https://stopcovid19.metro.tokyo.lg.jp/data/130001_tokyo_covid19_patients.csv';
  l_file_name := '130001_tokyo_covid19_patients.csv';
  update covid19_municipalities
  set content_blob = apex_web_service.make_rest_request_b(l_url, 'GET'),
      content_url = l_url,
      file_name = l_file_name,
      last_update_date = systimestamp
  where name = '東京都';
end;
```

設定されたURLが変更されなければ、データの更新は以下のSQLで実行できます。

```
update covid19_municipalities m
set m.content_blob = apex_web_service.make_rest_request_b(m.content_url, 'GET'),
    m.last_update_date = systimestamp
where m.name = '東京都';
```

保存したデータを使ったパース処理は以下になります。

```
select
    to_number(col001)  "No",
    to_number(col002) municipality_code,
    col003 prefecture_name,
    to_date(col005, 'YYYY-MM-DD') published_date,
    col008 patient_location,
    col009 patient_age,
    col010 patient_sex,
    to_number(col016) patient_left_hospital
from
    apex_data_parser.parse(
        p_content =>
          (select content_blob from covid19_municipalities where name = '東京都'),
        p_file_name => 'file_is.csv',
        p_skip_rows => 1
    ) where col001 is not null
```

東京都として与えている条件を変えることで、他の都道府県のデータを処理の対象にすることができます。

参考情報

CSV形式でデータを提供している都道府県

北海道、青森県、山形県、福島県、埼玉県、東京都、神奈川県、石川県、福井県、長野県、岐阜県、静岡県、三重県、岡山県、山口県、愛媛県、高知県、福岡県、長崎県、熊本県、大分県
(21自治体 - 7月20日現在)

Excel形式でデータを提供している都道府県

宮城県、栃木県、新潟県、富山県、山梨県、大阪府、兵庫県、奈良県(8自治体 - 7月20日現在)

WEBAPIを提供している都道府県

北海道、青森県、静岡県、愛媛県ではWEBAPIを呼び出すことで、提供しているCSVファイルのURLを取得できるようになっています。いくつかの都道府県では、更新がある度にファイル名を変更しているため、WEBAPIがあると便利です。上記の道と県が提供しているWEBAPIは、同じ形式のJSONドキュメントを返します。

返されるJSONドキュメント(対象部分のみ抜粋)です。


```
  {
    "id": 3827,
    "uuid": "f2805dd5-02cd-4e78-8aae-b81bb7a8a6b3",
    "revision_id": "bc508f0b-c8f4-4b21-b657-7c771df595f6",
    "name": "010006_hokkaido_covid19_patients.csv",
    "filename": "010006_hokkaido_covid19_patients.csv",
    "text": "「新型コロナウイルス感染症対策に関するオープンデータ項目定義書(Code for Japan)」に沿った形で情報をまとめた陽性患者属性データです。 \r\nhttps://www.code4japan.or
    "license": {
      "id": 1,
      "name": "表示(CC BY)",
      "uid": null,
      "state": "public",
      "created": "2018-03-07T18:35:30.492+09:00",
      "updated": "2018-03-07T18:35:30.500+09:00"
    },
```

```
      "rdf_iri": null,
      "rdf_error": null,
      "created": "2020-06-19T21:56:42.175+09:00",
      "updated": "2020-07-20T10:31:06.376+09:00",
      "download_url": "https://www.harp.lg.jp/opendata/dataset/1369/resource/3132/010006_hokkaido_covid19_patients.csv",
      "url": "https://www.harp.lg.jp/fs/3/8/2/7/_/010006_hokkaido_covid19_patients.csv",
      "format": "CSV"
    }
```


idが3827のリソースに含まれる、**download_url**と**filename**を取り出す必要があります。

APEX_WEB_SERVICEおよびAPEX_DATA_PARSERパッケージを使用することで、WEBAPIを呼び出してdownload_urlとfilenameを取得することができます。北海道では以下のコードで、データの取り込みを行うことができます。

```
declare
  l_url covid19_municipalities.content_url%type;
  l_file_name covid19_municipalities.file_name%type;
begin
  select col016, col019 into l_file_name, l_url from
    apex_data_parser.parse(
      p_content => apex_web_service.make_rest_request_b(
        p_url => 'https://www.harp.lg.jp/opendata/api/package_show?id=752c577e-0cbe-46e0-bebd-eb47b71b38bf',
        p_http_method => 'GET'
      ),
      p_file_type => 4
    )
  where col001 = '3827';
  update covid19_municipalities
  set content_blob = apex_web_service.make_rest_request_b(l_url, 'GET'),
      content_url = l_url,
      file_name = l_file_name,
      last_update_date = systimestamp
  where name = '北海道';
end;
```

APEX_WEB_SERVICEを使って取得したJSONドキュメントを、APEX_DATA_PARSERを使用してパースします。idはCOL001、filenameはCOL016、download_urlはCOL019として返されます。JSONドキュメントのパースでは、p_file_typeに4を指定します。

Excel形式の日付

宮城県など、日付データが**1900日付システム**による数値として取得される場合があります。

1900日付システムの説明(Microsoftのサイト)
https://docs.microsoft.com/ja-jp/office/troubleshoot/excel/1900-and-1904-date-system

以下の式を使って、Oracleの日付型に変換できます。

```
date'1900-01-01'  + Excel日付 - 2
```

Excelの日付は1900年1月1日を1とするので、date'1900-01-01' + Excel日付 - 1 ではと思ったのですが、-2します。-2をする理由は、Oracleでは1900年を平年として扱っていますが、Excelでは閏年として扱っているためです。厳密にいうと、1900年2月末日までは -1、それ以降は -2 となりますが、実用上はつねに -2 で問題ないでしょう。

続く

Yuji N. 時刻: 11:54

‹ 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