

日々是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。

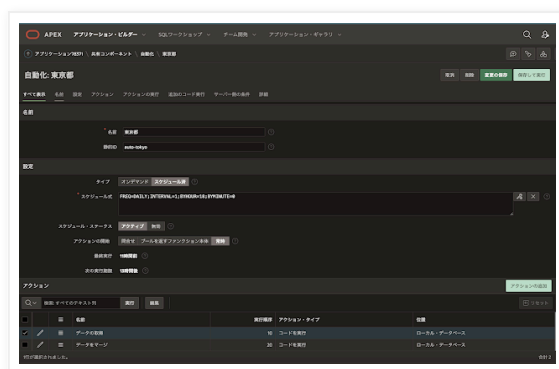
2020年11月11日 水曜日

APEX 20.2の自動化を使ってオープンデータの更新を行う

以前にOracle APEXの持つ[データ・ロードの機能を紹介する記事](#)を書きました。この中で、APIを使ってデータ・ロードを行う方法についても、実際にコードを示して解説しています。

Oracle APEX 20.2では、**共有コンポーネント**として**自動化(Automations)**が追加されました。この自動化によって、定期的、または必要に応じて、バックグラウンドで実行されるジョブを作成できるようになりました。これまでもOracle Databaseにて提供されているスケジューラー機能(DBMS_SCHEDULER)を使うことで、バックグラウンドでジョブを実行させることは可能でしたが、お世辞にも容易に利用できるとは言い難いものでした。

この機能の実際の使い方を、COVID-19の陽性患者データの取り込みを自動化することで紹介してみます。使用する表などは、元記事の方を参照してください。



定期的に行うPL/SQLスクリプト

元記事にある東京都のデータを取得するスクリプトは以下です。最初にネットワーク経由でCSVファイルを取得し、COVID19_MUNICIPALITIES表のCONTENT_BLOB列にバイナリ・データとして保存しています。(元記事からは、少々コードを改変しています)

```
begin
  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';
end;
```

次にバイナリ・データを解析して、COVID19_PATIENTS表に取り込みます。以下のコードによって行います。(こちら元記事からは、少々コードを改変しています)

```
merge into covid19_patients p
using
(
```

```

select
  to_number(col001) "No",
  to_number(col002) municipality_code,
  col003 prefecture_name,
  col004 city_name,
  to_date(col005,'YYYY-MM-DD') published_date,
  to_date(col007,'YYYY-MM-DD') onset_date,
  col008 patient_location,
  case
  when col009 = '-' then
    null
  when col009 = '不明' then
    null
  else
    col009
  end patient_age,
  case
  when col010 = '女' then
    '女性'
  when col010 = '不明' then
    null
  else
    col010
  end patient_sex,
  col011 patient_occupation,
  col012 patient_status,
  col013 patient_symptom,
  to_number(col014) patient_travel_history,
  to_number(col016) patient_left_hospital,
  col015 remark
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
minus
select
  "No", municipality_code, prefecture_name, city_name,
  published_date, onset_date,
  patient_location, patient_age, patient_sex, patient_occupation,
  patient_status, patient_symptom,
  patient_travel_history, patient_left_hospital, remark
from covid19_patients
where prefecture_name = '東京都'
) n
on (p."No" = n."No" and p.prefecture_name = n.prefecture_name)
when matched then
  update set
    p.city_name          = n.city_name,
    p.published_date     = n.published_date,
    p.onset_date         = n.onset_date,
    p.patient_location   = n.patient_location,
    p.patient_age        = n.patient_age,
    p.patient_sex        = n.patient_sex,
    p.patient_occupation = n.patient_occupation,
    p.patient_status     = n.patient_status,
    p.patient_symptom    = n.patient_symptom,
    p.patient_travel_history = n.patient_travel_history,
    p.patient_left_hospital = n.patient_left_hospital,
    p.remark             = n.remark

```

when not matched then

```
insert(  
  "No", municipality_code, prefecture_name, city_name,  
  published_date, onset_date,  
  patient_location, patient_age, patient_sex, patient_occupation,  
  patient_status, patient_symptom,  
  patient_travel_history, patient_left_hospital, remark  
)  
values  
(  
  n."No", n.municipality_code, n.prefecture_name, n.city_name,  
  n.published_date, n.onset_date,  
  n.patient_location, n.patient_age, n.patient_sex, n.patient_occupation,  
  n.patient_status, n.patient_symptom,  
  n.patient_travel_history, n.patient_left_hospital, n.remark  
);
```

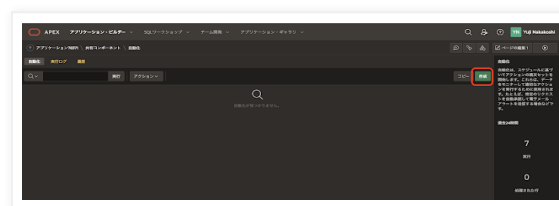
この処理を、定期的に夜間の実行されるようにします。

自動化の作成

共有コンポーネントの、アプリケーション・ロジックに含まれる**自動化**を開きます。



新規に自動化を作成するために、**作成**をクリックします。



自動化の作成フォームが開きます。

A screenshot of the '自動化の作成' (Create Automation) form. The form contains the following fields:

- 名前 (Name): 東京都
- タイプ (Type): オンデマンド (Selected), スケジュール済
- アクションの開始 (Action Start): 両合せ (Selected), 常時
- 実行スケジュール (Execution Schedule): 15分ごと (Selected), 正時, 毎日午前0時, カスタム
- 頻度 (Frequency): 日 (Selected), 毎時, 毎分
- 間隔 (Interval): 1 (Selected)
- 実行時間 (Execution Time): 9:00 (Selected), 午前0時, 午前6時, 正午, 午後6時

The form has '取消' (Cancel) and '作成' (Create) buttons at the bottom.

名前は任意です。ここでは東京都と入力しています。

タイプには2種類あります。オンデマンドはAPEX_AUTOMATIONパッケージのEXECUTEプロシージャを呼び出すことで、自動化を起動します。スケジュール済は、頻度、実行間隔と実行時間(時刻)を指定することで、自動化を定期的に起動します。

実行スケジュールには15分ごと、正時(毎時0分に実行)、毎日午前0時という簡易設定とカスタムがあります。一旦自動化を作成した後にも実行スケジュールの変更は可能で、その際には間隔ビルダーという、より柔軟なフォームによる設定が可能です。特に間隔ビルダーには頻度に週があり、曜日ごとの設定ができます。

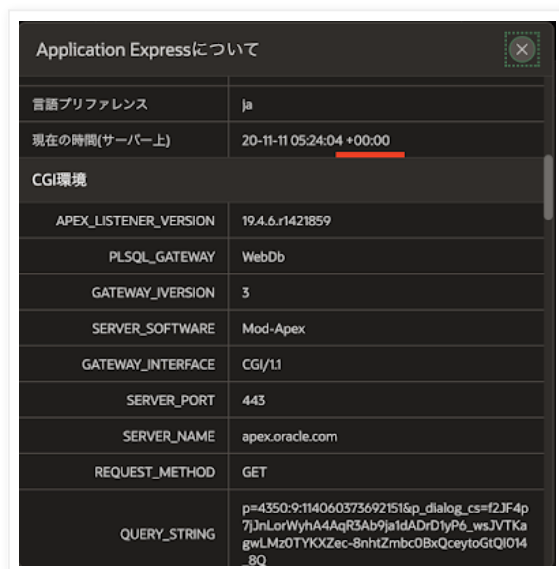


注意すべき点ですが、実行時間は必ずしも現地時間ではありません。サーバー上の現在の時刻を確認し、その時間帯での時刻を設定します。

アプリケーション・ビルダーの右上にある丸に？のアイコンをクリックしてメニューを開き、その中から情報を呼び出します。



Application Expressについてというダイアログの中に現在の時間(サーバー上)という項目があります。この中のタイムゾーン情報を確認します。クラウド上の環境はUTC (+00:00)、オンプレミスの環境では日本時間(+09:00)が一般的な設定でしょう。この記事で使っている環境はUTCでした。

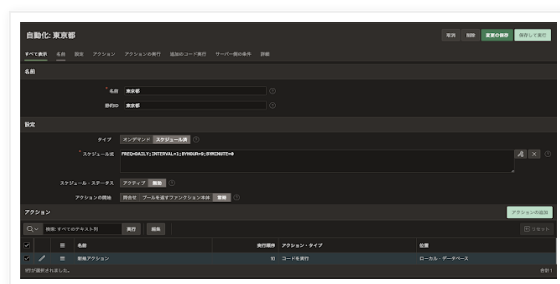


Application Expressについて	
言語プリファレンス	ja
現在の時間(サーバー上)	20-11-11 05:24:04 +00:00
CGI環境	
APEX_LISTENER_VERSION	19.4.6.r1421859
PLSQL_GATEWAY	WebDb
GATEWAY_IVERSION	3
SERVER_SOFTWARE	Mod-Apex
GATEWAY_INTERFACE	CGI/1.1
SERVER_PORT	443
SERVER_NAME	apex.oracle.com
REQUEST_METHOD	GET
QUERY_STRING	p=4350-9:114060373692151&p_dialog_cs=f2JF4p7JnLorWyhA4AgR3Ab9jaIdADrDtyP6_wsJVTka gwLMz0TYKXZec-8nhtZmbc0BxQceytoGtQlO148Q

毎日、日本時間の午後 1 8 時に自動化を起動するには、**頻度**として**日**、**間隔**は**1**、**実行時間**は**9:00**を指定します。

自動化の設定

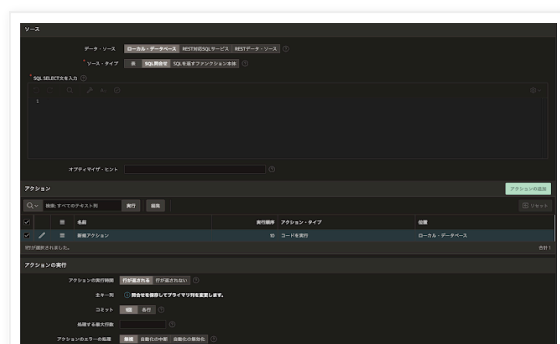
自動化が作成され、編集画面が開きます。



スケジュール・ステータスは、最初は**無効**です。これを**アクティブ**にし、**変更の保存**を行うことで、自動化が定期的に行われるようになります。**保存して実行**を行うと、タイプがスケジュール済の自動化であっても、定義されたアクションが即座に起動します。スケジュールの変更だけを行う場合は、変更の保存をクリックします。

アクションの開始には3種類あります。今回の例では**常時**を指定します。この指定では、スケジュールされた時刻になったら、定義されたアクションが実行されます。

それ以外の設定について説明します。



問合せを選ぶと、**ソース**として、**データ・ソース**と**ソース・タイプ**を設定します。**表**、**SQL問合せ**、もしくは、**SQLを返すファンクション本体**といった設定を行うことで、行の取得を行います。

起動されるアクションは、**アクションの実行時間**として、取得されたそれぞれの行に対してアクションが実行される、**行が返される**（取得した行数だけ、何回も実行される）か、または、行が選択されない場合に**1度だけ実行される**、**行が返されない**、を選択します。

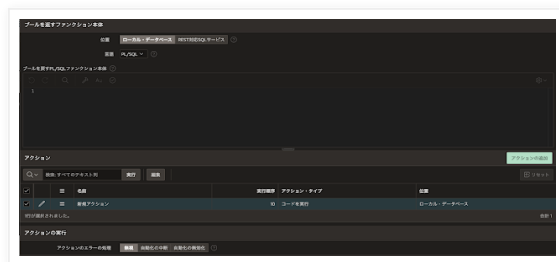
コミットでは、行ごとのアクションが実行された後にコミットするか(**各行**)、すべての行が処理された後で**1回**のみコミットするかを指定します。

処理する最大行数を指定した場合、最大行数に達すると自動化実行が停止し、自動化はログに「不完全」と記録されます。

アクションのエラー処理を無視、とした場合はエラーが発生しても、後続のアクションを継続して呼び出します。**自動化の中断**は、後続のアクションの呼び出しは行いませんが、次に実行する時刻

がきたら、また自動化は実行されます。**自動化の無効化**では、手動で再度アクティブにしない限り、自動化の実行は行われなくなります。

ブールを返すPL/SQLファンクション本体を選ぶと、真偽値を返すPL/SQLファンクション本体を記述するエディタが開きます。



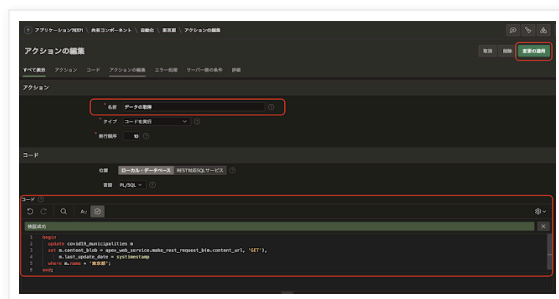
trueが戻された場合にアクションが1回、実行されます。

アクションの定義

自動化の作成時に、デフォルトでアクションがひとつ作成済みですので、最初にそれを編集します。新規アクションの**編集アイコン**（鉛筆）をクリックします。

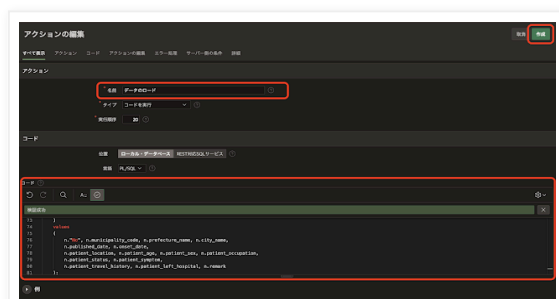


名前を**データの取得**とし、コードを入力した後、**変更の適用**をクリックします。

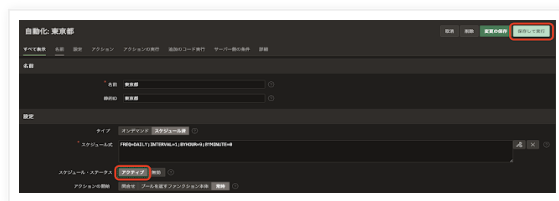


それぞれの行ごとに呼び出されるアクションの場合は、**:カラム名**として、バインド変数を使うことで処理対象となる行のデータに置き換えることができます。

次にアクションの追加をクリックし、名前を**データのロード**としてMERGE文を登録します。コードの入力後、**作成**をクリックします。



アクションの作成が完了したら、**スケジュール・ステータス**を**アクティブ**に切り替え、デバッグも兼ねて、**保存して実行**をクリックします。



一旦、自動化の一覧画面に戻り、**実行ログ**を確認します。



開始時刻(開始タイムスタンプ)、ステータス、成功した行、エラー行、メッセージを確認することができます。メッセージの数字はリンクになっており、クリックすることで出力されているメッセージを確認することができます。

メッセージの出力には、APEX_AUTOMATIONパッケージのLOG_INFO、LOG_WARN、LOG_ERRORプロシージャをアクションの中で使います。

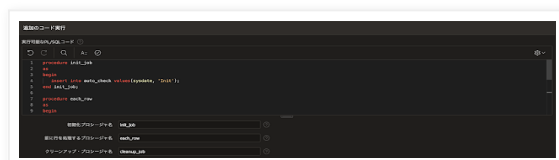
定期的なジョブの実行に関する設定は以上で完了です。



マニュアルの記載によると、自動化として設定されている処理は、必ずしも次回の実行の時刻ちょうどに実行されるわけではなく、数分程度、遅れる場合がある、とのこと。自動化はOracle Databaseのスケジューラにて実行されるコーディネータとしてのジョブより呼び出されるので、スケジューラのジョブと同じではありません。より厳密にこの時刻にジョブを呼び出したい、という場合は直接DBMS_SCHEDULERを使う必要があるでしょう。

追加のコード実行

高度な設定として、**追加のコード実行**があります。



一連のアクションが実行される前に、**初期化プロシージャ**が実行されます。アクションの開始が問合せの場合、それぞれの行に対してアクションが実行される前に、**前に行を処理するプロシージャ**が実行されます。全てのアクションが終了した後に**クリーンアップ・プロシージャ**が実行されます。

完

Yuji N. 時刻: 17:09

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](#)

Powered by Blogger.
