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1月4日月曜日

対話グリッドの列の計算方法について

Stefan Dobreさんによるこちらの記事 How to dynamically compute Interactive Grid Columns in #orclAPEX 20.2 の紹介です。

記事に記載されている実装を行うことで、理解してみました。色々と応用ができますし、同様の処理を大量の動的アクションで実装していた方は、ずっと分かりやすく書き換えることができるでしょう。

この機能は、Oracle APEXを20.2以降よりサポートされています。

以下が確認作業の順番です。

1. 表とアプリケーションの作成
2. CSVファイルのロード
3. 計算によって導出される列を、対話グリッドのSQLに追加
4. 対話グリッド上で、列の計算を行うコードを追加

１から２までは準備作業になります。Oracle APEXによるアプリケーション作成について経験のある方にとっては、冗長かもしれません。

表とアプリケーションの作成

サンプルの受注一覧のデータより、商品(**PRODUCT_NAME**)の価格(**PRICE**)と数量(**QUANTITY**)より、売り上げ(**REVENUE**)を計算します。売り上げは

REVENUE = PRICE * QUANTITY

として計算するため、表の列には含みません。

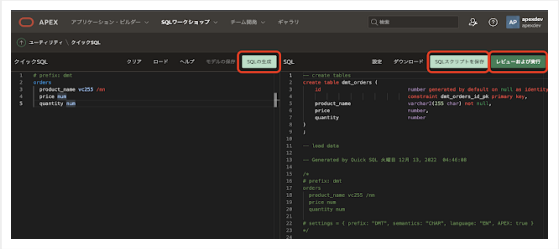
クイックSQLに与える設定は、以下とします。

```
# prefix: dmt
orders
```

```
product_name vc255 /nn
price num
quantity num
```

SQLワークショップより**クイックSQL**を開き、左側に上記のモデルを入力します。

SQLの生成、**SQLスクリプトを保存**、**レビューおよび実行**を順次実行します。

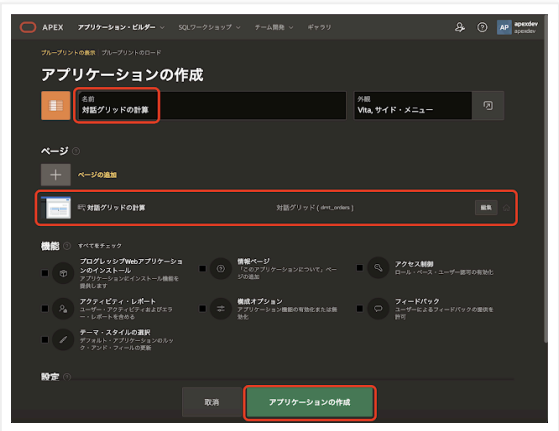


SQLスクリプトが開きます。DDLの変更は不要なので、そのまま**実行**します。確認画面が開くので、**即時実行**をクリックします。

表**DMT_ORDERS**が作成されます。続けて**アプリケーション作成**を実行します。

アプリケーション作成ウィザードが起動します。

作成するアプリケーションの**名前**は**対話グリッドの計算**とします。デフォルトで追加されている**ページをすべて削除**し（**編集**をクリックして**削除**を実行）、代わりに**ページの追加**をクリックして**対話グリッドを追加**します。

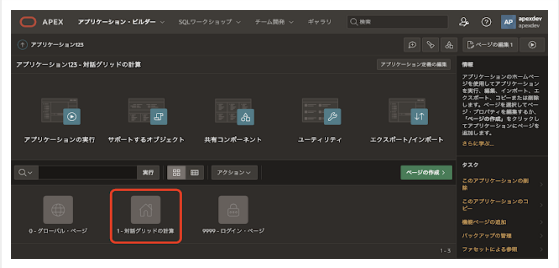


対話グリッドの**ページ名**も**対話グリッドの計算**とします。**表またはビュー**、**編集を許可**を選択し、**表またはビュー**として表**DMT_ORDERS**を選択します。

ページの追加をクリックします。

以上の設定を行い、**アプリケーションの作成**を実行します。

アプリケーションが作成されます。今回の作業はページ番号１の対話グリッドの計算のページに対して実施します。



CSVファイルのロード

ロードするCSVファイルを、以下よりダウンロードします。

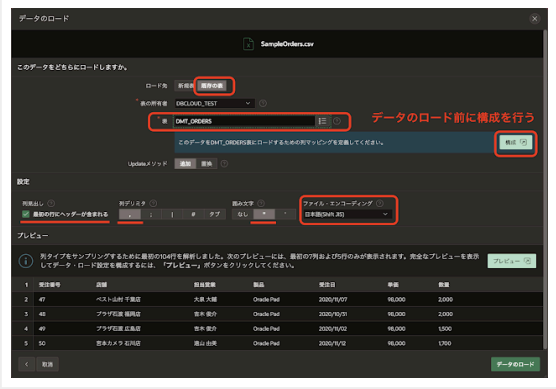
https://apex.oracle.com/pls/apex/japancommunity/r/simcontents/download?id=SampleOrders.csv

次に**SQLワークショップ**の**ユーティリティ**より、**データ・ワークショップ**を開きます。**データのロード**をクリックして開きます。

ファイルのアップロード（デフォルトで選択されています）より、**ファイルの選択**をクリックして、先ほどダウンロードした**SampleOrders.csv**を選択します。

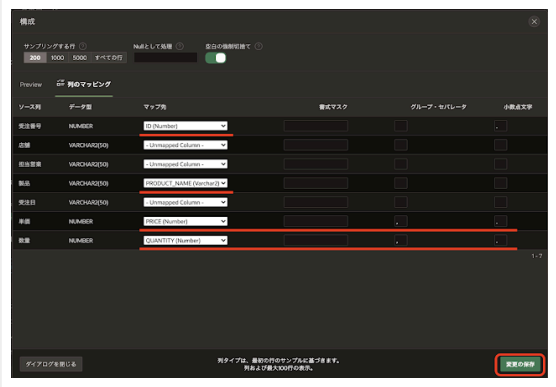
ロード先を**既存の表**、**表**としてクイックSQLで作成した表**DMT_ORDERS**、**ファイル・エンコーディング**は**日本語(Shift JIS)**を指定します。**最初の行にヘッダーが含まれる**は**チェック**する、**列デリミタ**の**,(カンマ)**、**囲み文字**の**"(ダブルクォート)**は、デフォルトのままとします。

データのロードを行う前に、**構成**をクリックします。

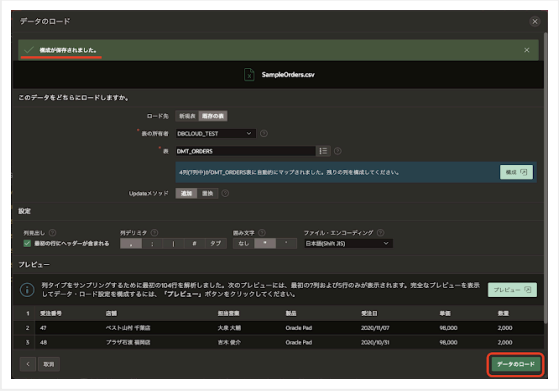


受注番号のマップ先として**ID(Number)**、**製品**は**PRODUCT_NAME(Varchar2)**、**単価**は**PRICE(Number)**、**数量**は**QUANTITY(Number)**を指定します。単価と数量には**グループ・セパレータ**として**,(カンマ)**、**小数点文字**として**.(ピリオド)**を指定します。私がテストしたときは、小数点文字のデフォルトが,(カンマ)になっていたので、設定変更は必須です。

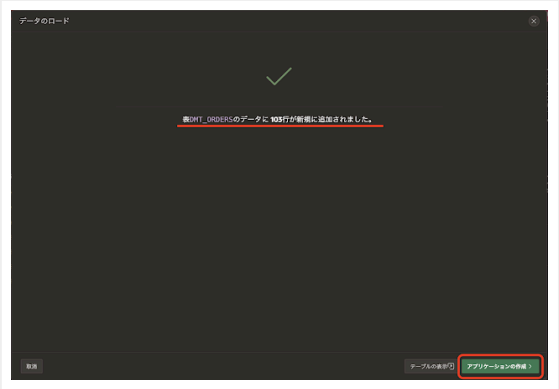
表DMT_ORDERSに含まれる４つの列をマップして、**変更の保存**を行います。



構成が保存されたので、**データのロード**を実行します。



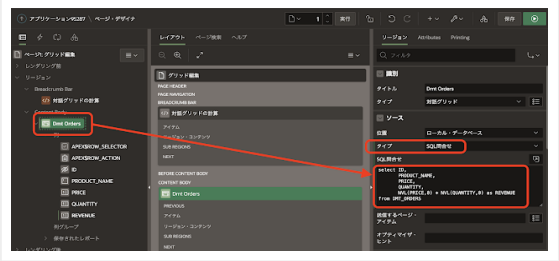
103行のデータがロードされていれば、ロードは成功です。続けて**アプリケーションの作成**を実行します。



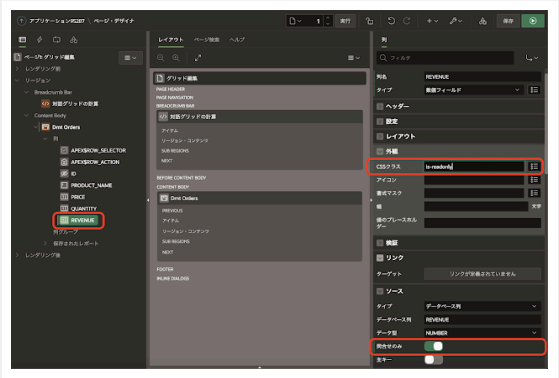
列REVENUEをSQLに追加

対話グリッドのページをページ・デザイナで開き、対話グリッドのリージョンを左ペインより選択します。**ソース**の**タイプ**を**SQL問合せ**に切り替え、SQL問合せを以下の列**REVENUE**を追加したものに更新します。

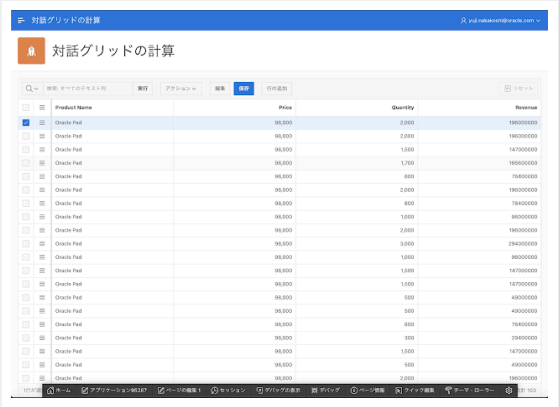
```
select ID,
      PRODUCT_NAME,
      PRICE,
      QUANTITY,
      NVL(PRICE,0) * NVL(QUANTITY,0) as REVENUE
from DMT_ORDERS
```



列**REVENUE**が追加されます。元記事によると、今回のソリューションは編集可能な列にのみ適用可能なので、タイプを表示のみに設定することは不可です。しかし、列REVENUEは計算された結果でありユーザーからの入力は禁止したいので、**外観のCSSクラス**に**is-readonly**を指定します。また、データベースのアップデート操作に含まれないよう**問合せのみ**を**ON**にします。



変更を保存して、ページを実行します。



この状態で**列Price、Quantityを変更**して、**保存**をクリックすると**列Revenueに計算結果**が反映されることが確認できます。

次に、保存をクリックせず、PriceやQuantityを変更するとすぐにRevenueが更新される実装を行います。

対話グリッド上の列の計算

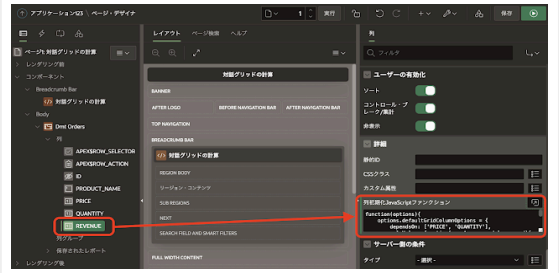
Oracle APEX 20.2から利用可能になったcalcValueとdependsOnの機能を使用します。

列REVENUEのプロパティである**詳細**の**列初期化JavaScriptファンクション**に以下を記述します。

```
function(options){
    options.defaultGridColumnOptions = {
        dependsOn: ['PRICE', 'QUANTITY'],
        calcValue: function(argsArray, model, record){
            const price = parseInt(model.getValue(record, 'PRICE').replaceAll(",","") || 0);
            const quantity = parseInt(model.getValue(record, 'QUANTITY').replaceAll(",","") ||
            if(isNaN(price) || isNaN(quantity)){
                return 'error';
            } else {
                return price * quantity;
            }
        }
```

```
    };
    return options;
}
```

calcvalue-interactive-grids.js hosted with ❤️ by GitHub view raw



dependsOnとして、この列REVENUEが依存している列を指定します。REVENUEはPRICEとQUANTITYの積なので、dependsOnはPRICEとQUANTITYになります。

dependsOnで指定された列、PRICEまたはQUANTITYが変更されたときに呼び出されるファンクションを**calcValue**として記述します。戻り値が列REVENUEの値になります。

以上のコードを設定して、期待した動作になるか確認してみましょう。列REVENUEを計算しているJavaScriptのコードの部分については、理解が難しいところはないかと思います。

おまけ：書式マスクについて

数値列である列PRICEとQUANTITYに以下のような書式マスクの設定がされていると、３桁ごとに区切り文字として,(カンマ)が含まれます。

999G999G999G999G999G999G999G999G999G990

このままではparseInt()で適切な整数として変換されないので、replaceAll(",","")でカンマを取り除いています。

列PRICEやQUANTITYを変更すると列REVENUEが更新されますが、この計算結果には,(カンマ)が含まれません。一旦、対話グリッドを保存すると書式マスクが適用され、,(カンマ)が表示されます。

対話グリッドの計算結果としても,(カンマ)を含めるには、Intl.NumberFormatを使って以下のように記述することで対応できます。

```
function(options){
    const formatter = Intl.NumberFormat("ja-JP");
    options.defaultGridColumnOptions = {
        dependsOn: ['PRICE', 'QUANTITY'],
        calcValue: function(argsArray, model, record){
            const price = parseInt(model.getValue(record, 'PRICE').replaceAll(",","") || 0);
            const quantity = parseInt(model.getValue(record, 'QUANTITY').replaceAll(",","") ||
            if(isNaN(price) || isNaN(quantity)){
                return 'error';
            } else {
                return formatter.format(price * quantity);
```

```
            }
        }
    };
    return options;
}
```

calcvalue-interactive-grids-comma.js hosted with ❤ by GitHub view raw

以上で、対話グリッドの列の計算方法の紹介は終了です。

今回作成したアプリケーションのエクスポートを以下に置きました。
https://github.com/ujnak/apexapps/blob/master/exports/calcvalue-interactive-grids.zip

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: 15:29

共有

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