日日是Oracle APEX

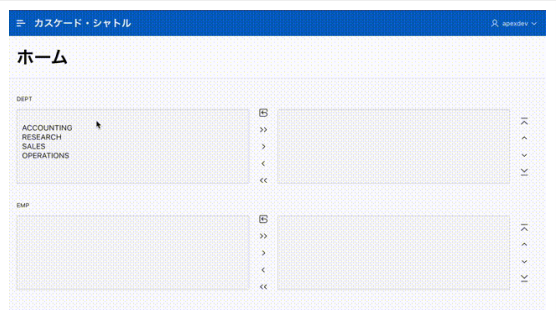
Oracle APEXを使った作業をしていて、気の付いたところを忘れないようにメモをとります。

2022年12月16日金曜日

カスケードLOVのソース定義について

２つのページ・アイテムがあり、それぞれタイプがシャトルとなっています。このページ・アイテムには親子関係があり、親となるシャトルで選択した値に基づき、子のシャトルで選択できる値が決まります。

サンプル・データセットの**EMP/DEPT**をデータ・ソースとして使い、以下のようなページを作成します。

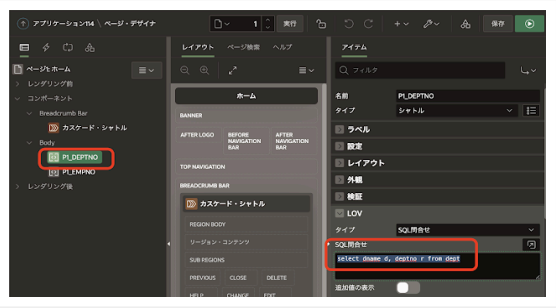


基本の実装

親のシャトルが**P1_DEPTNO**として作成されています。表**DEPT**から**DEPTNO**を複数選択します。

LOVの**SQL問合せ**は以下になります。

select dname d, deptno r from dept



従業員を選択する子であるシャトルは**P1_EMPNO**として作成されています。

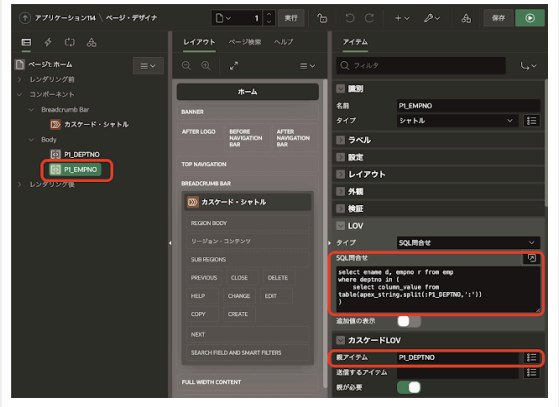
LOVの**SQL問合せ**は以下になります。**カスケードLOV**の**親アイテム**として**P1_DEPTNO**を指定します。**親アイテム**として指定したアイテムは必ず送信されるため、**送信されるアイテム**に指定する必要はありません。

```sql
select ename d, empno r from emp
where deptno in (
    select column_value from table(apex_string.split(:P1_DEPTNO,':'))
)
```

cascade_lov_child_source.sql hosted with ❤️ by GitHub view raw

ページ・アイテムに複数の値が設定される場合、それぞれの値は':'（コロン）で区切られます。そのため、P1_DEPTNOの値をapex_string.splitを呼び出して分割しています。

ページ・アイテムP1_DEPTNOのシャトルで選択した部門の従業員に限り、P1_EMPNOのシャトルで選択できるようになります。



ここまでが基本の実装です。

SQLを記述できない場合

従業員の表がそれぞれ部門で分割されているケースを想定します。

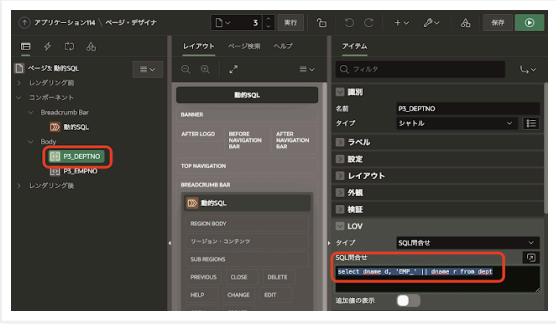
create table emp_accounting as select * from emp where deptno = 10;
create table emp_research as select * from emp where deptno = 20;
create table emp_sales as select * from emp where deptno = 30;
create table emp_operations as select * from emp where deptno = 40;

部門ごとに４つの表が作られています。**EMP_ACCOUNTING**、**EMP_RESEARCH**、**EMP_SALES**、**EMP_OPERATIONS**です。親となるシャトルではこれらの表を選択し、子となるシャトルは選択された表から従業員を選択します。

親のシャトルの**SQL問合せ**は以下に変わります。

select dname d, 'EMP_' || dname r from dept

ページ・アイテム**P3_DEPTNO**はコロンで区切られた表名を複数持ちます。

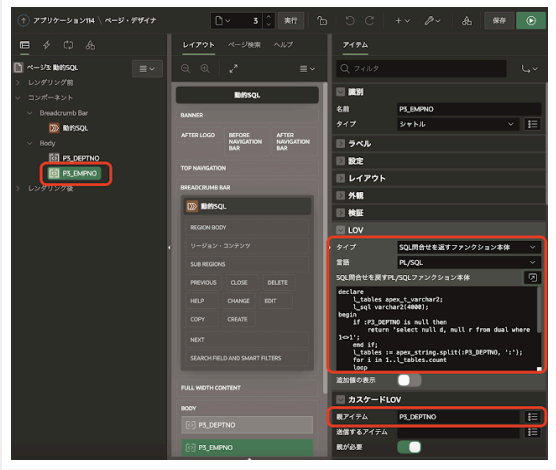


子のシャトルでは、**LOVの**タイプを**SQL問合せを返すファンクション本体**に変更し、**SQL問合せを戻すPL/SQLファンクション本体**として以下を記述します。ファンクションの評価時はページ・アイテムP3_DEPTNOはnullになります。評価に失敗するため、０行を返すダミーのSELECT文を戻しています。

```
declare
    l_tables apex_t_varchar2;
    l_sql varchar2(4000);
begin
    if :P3_DEPTNO is null then
        return 'select null d, null r from dual where 1<>1';
    end if;
    l_tables := apex_string.split(:P3_DEPTNO, ':');
    for i in 1..l_tables.count
    loop
        if l_sql is not null then
            l_sql := l_sql || ' union ';
        end if;
        l_sql := l_sql || 'select ename d, empno r from ' || dbms_assert.enquote_name(l_tables(
    end loop;
    return l_sql;
end;
```

generate_cascade_lov_source.sql hosted with ❤️ by GitHub view raw

カスケードLOVは基本と同じ設定になります。



動的にSELECT文を作成できない場合

動的にSQLを作る場合でも、最終的には実行可能なSELECT文が生成される必要があります。SELECT文を作るのが困難なケース（RESTサービスを呼び出すなど）では、パイプライン表関数を作成することで対応できる場合もあります。

今回の例を実装すると、以下のようになります。

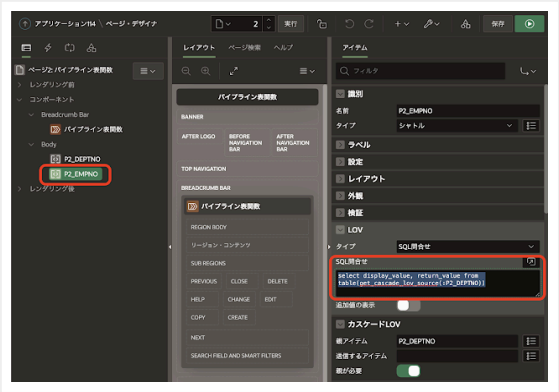
```
create or replace type t_cascade_lov_source_row as object(
    display_value varchar2(256)
    ,return_value number
);
/

create or replace type t_cascade_lov_source_table as table of t_cascade_lov_source_row;
/

create or replace function get_cascade_lov_source(
    p_table_names in varchar2
)
return t_cascade_lov_source_table pipelined
is
    type t_ref_cursor is ref cursor;
    refc t_ref_cursor;
    l_tables apex_t_varchar2;
    l_sql varchar2(4000);
    l_display_value varchar2(256);
    l_return_value  number;
begin
    l_tables := apex_string.split(p_table_names, ':');
    for i in 1..l_tables.count
    loop
        l_sql := 'select ename, empno from ' || dbms_assert.enquote_name(l_tables(i));
        open refc for l_sql;
        loop
            fetch refc into l_display_value, l_return_value;
            exit when refc%NOTFOUND;
            pipe row( t_cascade_lov_source_row( l_display_value, l_return_value ));
        end loop;
        close refc;
    end loop;
end get_cascade_lov_source;
/
```

get_cascade_lov_source.sql hosted with ❤ by GitHub　　view raw

作成したパイプライン表関数**get_cascade_lov_source**を使用した**SQL問合せ**は以下になります。

select display_value, return_value from table(get_cascade_lov_source(:P2_DEPTNO))

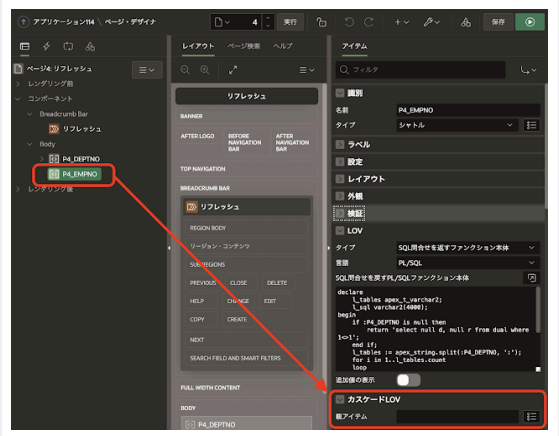
パイプライン表関数による実装の場合、表示値と戻り値を返せばよいので実際のデータ・ソースが表である必要はありません。



カスケードLOVの親アイテムを指定できない場合

あまり無いとは思いますが、カスケードLOVの親アイテムを設定できない場合を考えてみます。

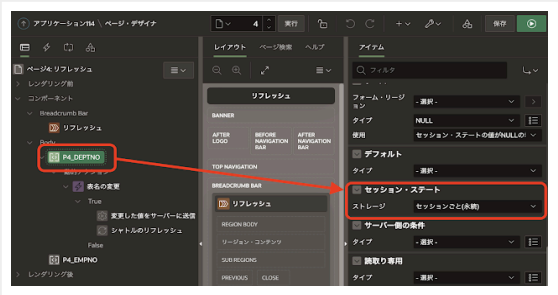
この場合、親となるページ・アイテムが変更されたときに、子となるページ・アイテムをリフレッシュする必要があります。



ページ・アイテムの**変更**イベントでページ・アイテムの**リフレッシュ**を実行します。このTRUEアクションは、一般的なリフレッシュの設定です。

子アイテムの**LOV**の**SQL問合せ**、または、**ファンクション本体**に親アイテムであるページ・アイテムがバインド変数として使用されている場合、カスケードLOVの親アイテムを指定せずに、送信するページ・アイテムを指定する方法はありません。

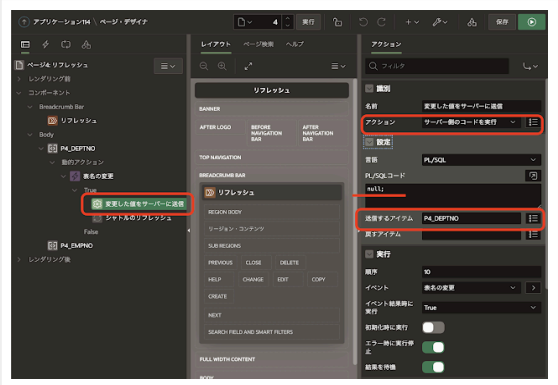
このため、親となるページ・アイテムの**セッション・ステート**の**ストレージ**を**セッションごと（永続）**に切り替えます。



TRUEアクションを作成し、子であるページ・アイテムをリフレッシュする前に配置します。

アクションに**サーバー側のコードを実行**を選択します。**PL/SQLコード**は不要なので**null;**とします。**送信するアイテム**として**親となるページ・アイテム**を指定します。

セッション・ステートの定義は**セッション（永続）**になっているため、送信されたページ・アイテムの値はデータベースに保存されます。子アイテムのソースとなるコードが実行される際に、サーバーに保存されている値がバインド変数に割り当てられるため、親アイテムの値が反映された結果が得られます。



今回の記事は以上です。

検証に使用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/cascade-shuttle-lov-source.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: 12:18

共有

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