Trinamix

Dynamic Query in Data Model-Oracle Fusion

Contents

Abstract 2

Issues**:** 2

Key Benefits: 2

Customized Solution: 3

Abstract

This blog offers an in-depth guide on crafting a data model from dynamic tables. In scenarios like supply planning where dynamic tables are generated, a solution is imperative. Oracle presents a procedural approach to building a data model, providing a viable solution for such requirements.

Issues:

i) Once deployed in the production environment, adjustments to the BIP reports' code are necessary for the actual DYD tables.

ii) Changes in plan names initiated by the business necessitate modifications in the code, which may sometimes result in delays in meeting the requirements.

Key Benefits:

i) With the new tailored process in place, code changes are unnecessary as DYD tables will be generated dynamically.

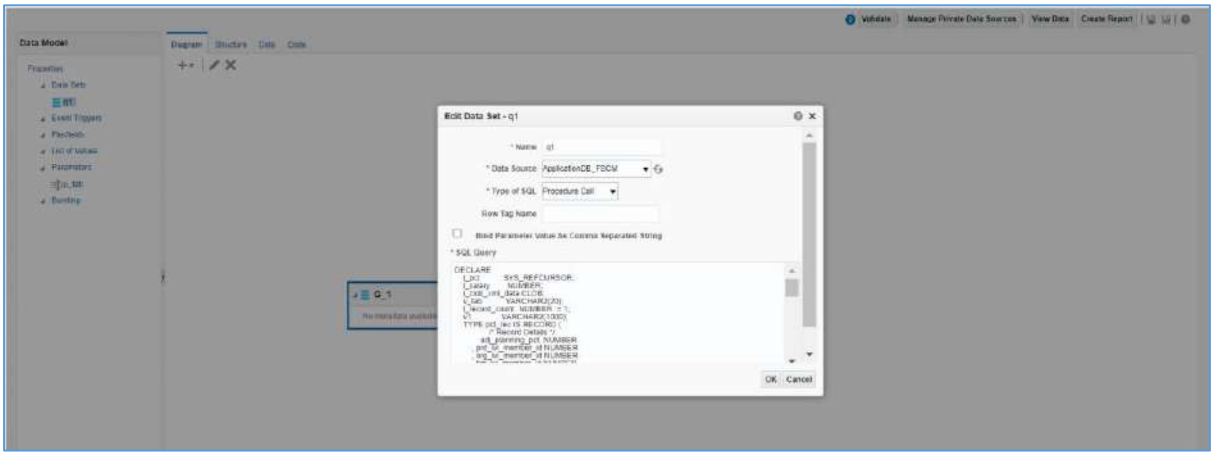
Existing Process/Oracle Standard Process:

Most Oracle developers write code that may later require modifications.

Below is an outline of the steps for the customized solution that will generate a dynamic query.

Customized Solution:

Step1. Create a Data Model (Type of SQL - Procedural Call)



Edit Data Set - q1

* Name: q1

* Data Source: ApplicationDB_FSCM

* Type of SQL: Procedure Call

Row Tag Name

□ Bind Parameter Value As Comma Separated String

* SQL Query

```
DECLARE
   l_pct          SYS_REFCURSOR;
   l_salary        NUMBER;
   l_clob_xml_data CLOB;
   v_tab          VARCHAR2(20);
   l_record_count  NUMBER := 1;
   v1             VARCHAR2(1000);
   TYPE pct_rec IS RECORD (
          /* Record Details */
       adj_planning_pct  NUMBER
     , prd_lvl_member_id NUMBER
     , org_lvl_member_id NUMBER
```

Step2.

This data model will have two sections:

- ✓ In the **DECLARE** section, we need to define the variables and the **function** to execute the dynamic query.
- ✓ To get the table name, it could be passed in parameter or derive from the query into the variable.
- ✓ In the **BEGIN** section, the query could be executed dynamically by calling the function - func_cur
- ✓ The results need to captured in a BLOB variable l_clob_xml_data
- ✓ Finally the BLOB variable need to be opened in a cursor - :xdo_cursor

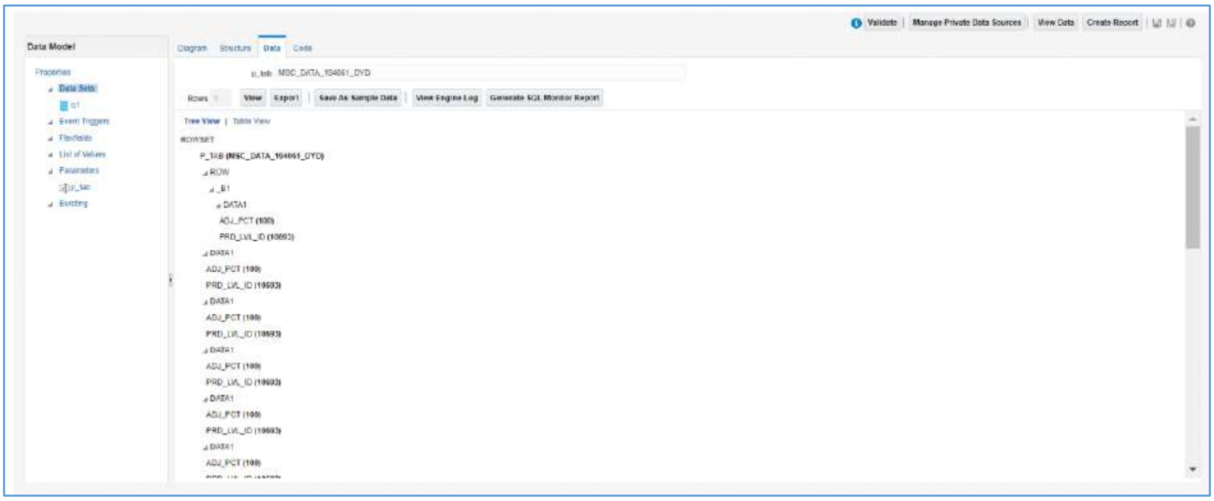
Below is the code snippet:

```
DECLARE

l_pct          SYS_REFCURSOR;
   l_salary       NUMBER;
   l_clob_xml_data CLOB;
   v_tab          VARCHAR2(20);
   l_record_count  NUMBER := 1;
   v1             VARCHAR2(1000);
   TYPE pct_rec IS RECORD (
                /* Record Details */
         adj_planning_pct  NUMBER
       , prd_lvl_member_id NUMBER
       , org_lvl_member_id NUMBER
       , tim_lvl_member_id NUMBER
   );
```

```
  TYPE pct_table IS
    TABLE OF pct_rec INDEX BY BINARY_INTEGER;
  pct_record      pct_rec;
  pct_data        pct_table;

  FUNCTION func_cur (
    query_in IN VARCHAR2
  ) RETURN SYS_REFCURSOR IS
    l_return SYS_REFCURSOR;
  BEGIN
    OPEN l_return FOR query_in;

    RETURN l_return;
  END func_cur;

BEGIN
 v_tab :=   :p_tab; --  'MSC_DATA_194061_DYD';

SELECT DISTINCT
   mpt.pds_table_name dyd_table_name
into   v_tab
FROM
  fusion.msc_measures_b  mmb
 , fusion.msc_measures_tl mmt
 , msc_measures_vl       v
 , msc_plan_definitions   mpd
 , msc_plan_tables        mpt
WHERE
    1 = 1
  AND mmb.measure_id = mmt.measure_id
  AND mmb.measure_id = v.measure_id
  AND mmb.measure_code = v.measure_code
  AND mmt.language = 'US'
  AND mmt.name IN ( 'Measure Name' )
  AND TRIM(mpd.compile_designator) = 'Plan Name'
  AND mpt.plan_id = mpd.plan_id
  AND mpt.entity_id = v.granularity_id
  AND mpt.entity_type IN ( 'DATA' )
ORDER BY
  mmt.name
 , v.granularity_id
 , mpt.entity_type
  v1 := 'SELECT  ADJ_PLANNING_PCT , prd_lvl_MEMBER_ID  , org_lvl_MEMBER_ID,
tim_lvl_member_id from '
      || v_tab – this value can be derived from a quuery or could be passed as a
parameter.
      || ' WHERE cto_lvl_member_id = 53007 AND cus_lvl_member_id = 3331887';
  l_pct := func_cur(v1); --  here the sql is being executed
  l_clob_xml_data := empty_clob();
  FETCH l_pct
  BULK COLLECT INTO pct_data;
  FOR i IN 1..pct_data.count LOOP
    l_clob_xml_data := l_clob_xml_data||to_clob('<DATA1>');
    l_clob_xml_data := l_clob_xml_data
              || to_clob('<ADJ_PCT>'
                  || pct_data(i).adj_planning_pct
                  || '</ADJ_PCT>');

    l_clob_xml_data := l_clob_xml_data
              || to_clob('<PRD_LVL_ID>'
                  || pct_data(i).prd_lvl_member_id
                  || '</PRD_LVL_ID>');

    l_clob_xml_data := l_clob_xml_data||to_clob('</DATA1>');
  END LOOP;

  CLOSE l_pct;

 --  l_clob_xml_data := v1;
 -- l_clob_xml_data :=  '<RECORD_COUNT>' || '2'|| '</RECORD_COUNT>';
  OPEN :xdo_cursor FOR SELECT
              l_clob_xml_data
             FROM
              dual;
    --   OPEN :xdo_cursor FOR SELECT  model_id from cz_rules where rule_id=1000;
END;
```

Step3.

Execute the Data Model: (In this case, it is executed using the table name as a parameter)

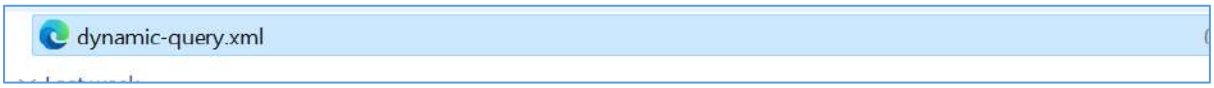


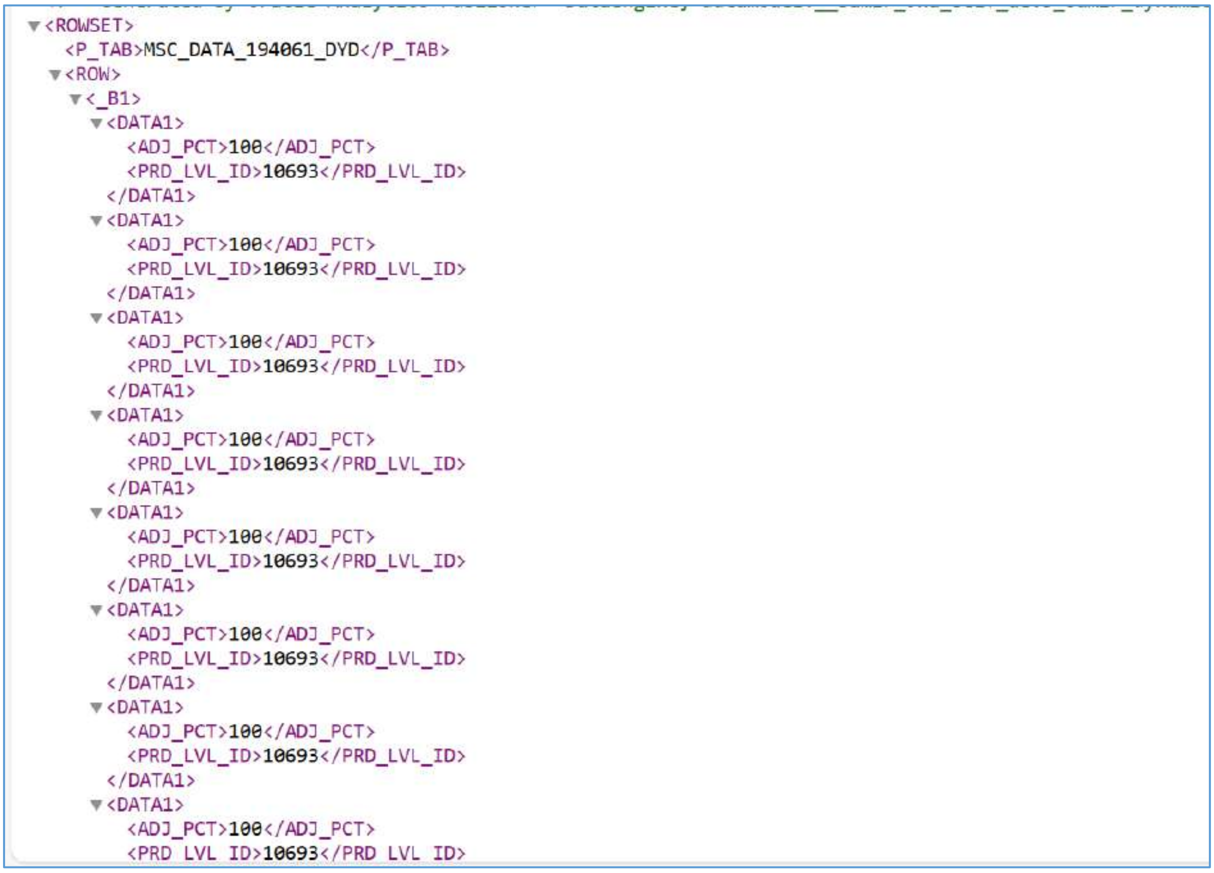
p_tab MSC_DATA_194061_DYD

```
ROWSET
  P_TAB (MSC_DATA_194061_DYD)
    ROW
      _B1
        DATA1
          ADJ_PCT (100)
          PRD_LVL_ID (10693)
    DATA1
    ADJ_PCT (100)
    PRD_LVL_ID (10693)
    DATA1
    ADJ_PCT (100)
    PRD_LVL_ID (10693)
    DATA1
    ADJ_PCT (100)
    PRD_LVL_ID (10693)
    DATA1
    ADJ_PCT (100)
    PRD_LVL_ID (10693)
    DATA1
    ADJ_PCT (100)
```

Step4.

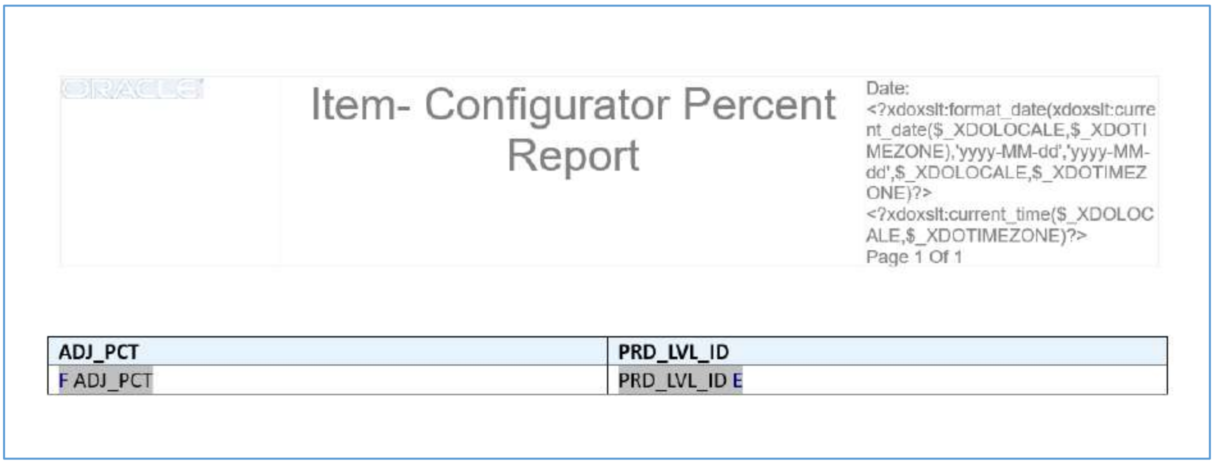
Export the Data as XML:


dynamic-query.xml



```
<ROWSET>
  <P_TAB>MSC_DATA_194061_DYD</P_TAB>
  <ROW>
    <_B1>
      <DATA1>
        <ADJ_PCT>100</ADJ_PCT>
        <PRD_LVL_ID>10693</PRD_LVL_ID>
      </DATA1>
      <DATA1>
        <ADJ_PCT>100</ADJ_PCT>
        <PRD_LVL_ID>10693</PRD_LVL_ID>
      </DATA1>
      <DATA1>
        <ADJ_PCT>100</ADJ_PCT>
        <PRD_LVL_ID>10693</PRD_LVL_ID>
      </DATA1>
      <DATA1>
        <ADJ_PCT>100</ADJ_PCT>
        <PRD_LVL_ID>10693</PRD_LVL_ID>
      </DATA1>
      <DATA1>
        <ADJ_PCT>100</ADJ_PCT>
        <PRD_LVL_ID>10693</PRD_LVL_ID>
      </DATA1>
      <DATA1>
        <ADJ_PCT>100</ADJ_PCT>
        <PRD_LVL_ID>10693</PRD_LVL_ID>
      </DATA1>
      <DATA1>
        <ADJ_PCT>100</ADJ_PCT>
        <PRD_LVL_ID>10693</PRD_LVL_ID>
      </DATA1>
      <DATA1>
        <ADJ_PCT>100</ADJ_PCT>
        <PRD_LVL_ID>10693</PRD_LVL_ID>
```

Step5. Create the RTF according to the structure of the data model:

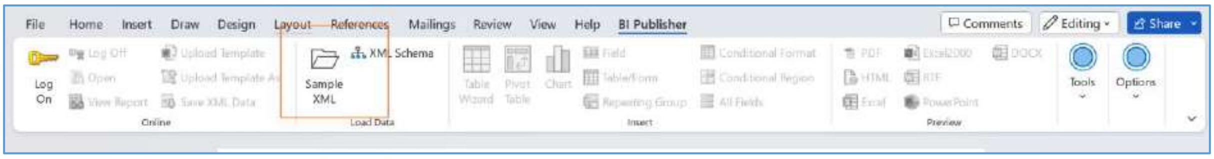


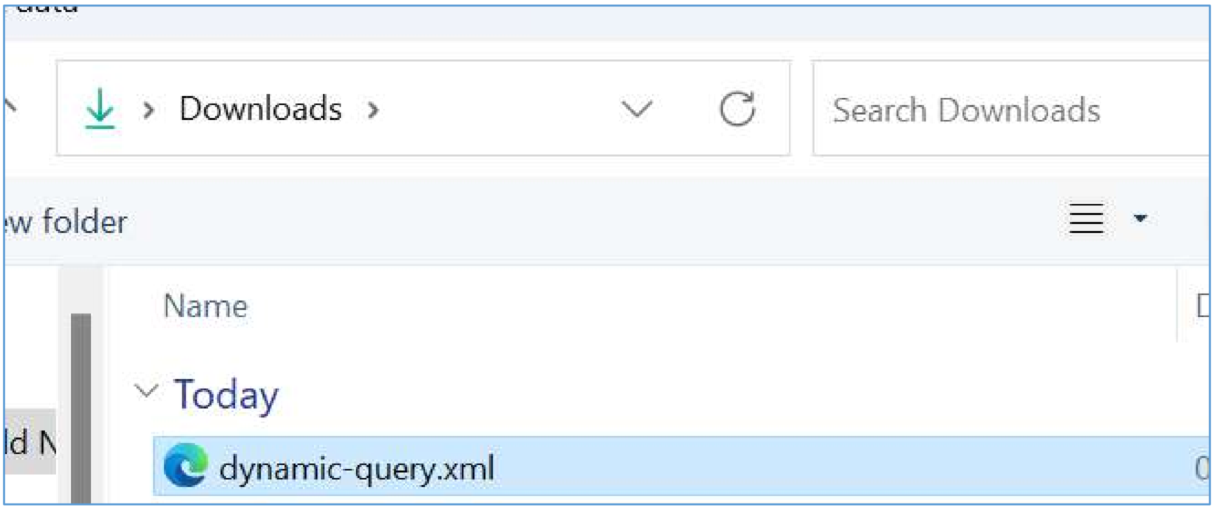
Item- Configurator Percent Report

Date: <?xdoxslt:format_date(xdoxslt:current_date($_XDOLOCALE,$_XDOTIMEZONE),'yyyy-MM-dd','yyyy-MM-dd',$_XDOLOCALE,$_XDOTIMEZONE)?> <?xdoxslt:current_time($_XDOLOCALE,$_XDOTIMEZONE)?>
Page 1 Of 1

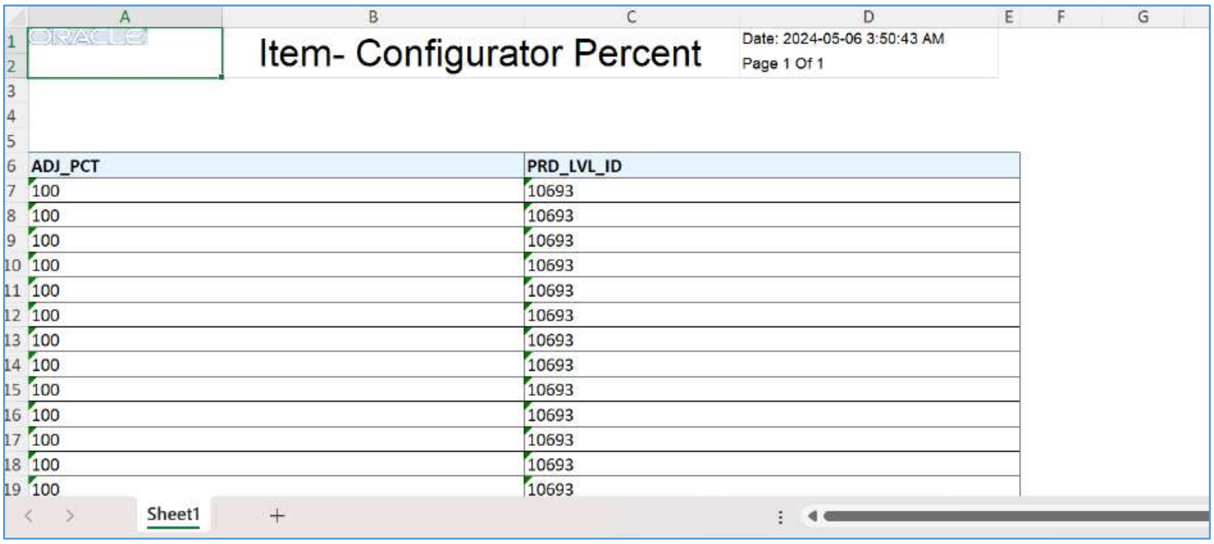
ADJ_PCT	PRD_LVL_ID
F ADJ_PCT	PRD_LVL_ID E

Export the data model:





Sample Output:



Item- Configurator Percent

Date: 2024-05-06 3:50:43 AM

Page 1 Of 1

ADJ_PCT	PRD_LVL_ID
100	10693
100	10693
100	10693
100	10693
100	10693
100	10693
100	10693
100	10693
100	10693
100	10693
100	10693
100	10693
100	10693

About Author: -

Samir has over over 13 years of IT experience, currently working at Trinamix Systems Pvt. Ltd since Sep’23. He has worked at LTI Mindtree since (Feb,22 to Aug23) and TCS from March 2010 to February 2022. His role includes as a Solution Architect (Techno-Functional Consultant) specialising in Fusion Cloud - SCM/Finance, I excel in various domains such as Business Requirement Gathering, Gap & Risk Analysis, System Design, Application Software Development, Implementation, and Application Testing.