## CE 函数介绍

CE function，即Calculation Engine Plan Operators。因为类似函数的形式，包含输入参数和输出参数，并且封装了对数据的转化的操作，常被称作CE函数，或者CE function。CE function 被用于SQLScript 存储过程中，使用CE function的存储过程可以直接在calculation engine上进行操作。

一般的CE function都有与其相对应的SQL 代码的查询的实现方式，两者的使用二选其一，常用的CE function简介数据源的访问操作系列：将列存储表格或者列视图作为数据源与表变量相绑定。

## 1.CE\_COLUMN\_TABLE

这个函数使用一个已经存在的列存储表格作为数据源，选取其全部或者部分列，返回相应内容绑定到一个表变量上。     例如：选取表PRODUCT 的NAME,COLOR,PRICE三列，绑定到表变量product1上。   
  
product1 = CE\_COLUMN\_TABLE("PRODUCT",["NAME","COLOR","PRICE"]);    
  
     使用SQL语句表示为：  
  
product1 = select"NAME","COLOR","PRICE"from"PRODUCT";

## 2.CE\_JOIN\_VIEW

  这个函数使用一个已经存在的连接视图（即Attribute View，属性视图）作为数据源，选取其全部或者部分列，返回相应内容绑定到一个表变量上。  
例如：选取属性视图 ATTR\_PRODUCT的PRODUCT\_NAME,CATEGORY\_NAME两列，绑定到表变量product2上。

product2 = CE\_JOIN\_VIEW("ATTR\_PRODUCT",["PROCUCT\_NAME","CATEGORY\_NAME"]);    
  
     使用SQL语句表示为：  
  
product2 = select"PROCUCT\_NAME","CATEGORY\_NAME"from"ATTR\_PRODUCT";  

## 3.CE\_OLAP\_VIEW

这个函数使用一个已经存在的OLAP视图（即Analytical  View，分析视图）作为数据源，选取一些关键值或者维度作为参数，按这些维度分度并求对应的聚合值，返回相应内容绑定到一个表变量上。  
     例如：选取分析师图ANAL\_SALE作为数据源，按CATEGORY\_NAME分组，并求对应每组AMOUNT的和，返回结果绑定到表变量olap\_view上。  
  
  
olap\_view = CE\_OLAP\_VIEW("ANAL\_SALE", ["CATEGORY\_NAME", SUM("AMOUNT")]);    
  
     使用SQL语句表示为：  
  
olap\_view = select CATEGORY\_NAME, SUM(AMOUNT) FROM  
"ANAL\_SALE"  
GROUP  
BY CATEGORY\_NAME;   

## 4.CE\_CALC\_VIEW

这个函数使用一个已经存在的计算视图（即Calculation View视图）作为数据源，选取其全部或部分属性，返回相应内容绑定到一个表变量上。  
     例如：选取计算视图CALC\_SALE的QUANTITY和AMOUNT属性，绑定到表变量cal上。  
  
cal = CE\_CALC\_VIEW("CALC\_SALE",["QUANTITY","AMOUNT"]);    
  
     使用SQL语句表示为：  
  
cal = select  
"QUANTITY","AMOUNT"  
from  
"CALC\_SALE";    
关系运算系列：可以直接在calculation engine上执行关系运算。

## 5.CE\_JOIN

这个函数可以得出给出的两个表变量通过某些属性相关联的内连接的结果集。对于相关联的两个属性，结果中只包含一个，因此要求必须有相同的属性名，否则提前进行重命名。同样，可以选择在结果中显示内连接结果的全部或部分属性。     例如:表变量:product和:category通过属性”CATEGORYID”进行内连接，并且结果中保留"PRODUCTID","PROJECT\_NAME","CATEGORYID","NAME"四个属性。  
  
product3 = CE\_JOIN(:project,:category,["CATEGORYID"],["PRODUCTID","PROJECT\_NAME","CATEGORYID","NAME"]);    
  
     使用SQL语句表示为：  
  
product3= select P."PRODUCTID",P."PROJECT\_NAME",P."CATEGORYID",C."NAME"  
from :project as P inner join :category  as C on P."CATEGORYID"=C."CATEGORYID";   

## 6.CE\_LEFT\_OUTER\_JOIN

使用方式与CE\_JOIN相同，实现左外连接的功能。

## 7.CE\_RIGHT\_OUTER\_JOIN

使用方式与CE\_JOIN相同，实现右外连接的功能。

## 8.CE\_PROJECTION

这个函数选取表变量中的一列或者多列，可以对每列进行重命名，计算表达式或者使用过滤条件。     例如：选取表变量:product中按条件"PRICE">50筛选并保留的"PRODUCTID","PRICE","NAME"三列，将”NAME”重命名为”CATEGORYNAME”。  
  
product4 = CE\_PROJECTION(:product,["PRODUCTID","PRICE","NAME"  
AS  
"CATEGORYNAME"],'"PRICE">50');    
  
     使用SQL语句表示为：  
  
product4 = select  
"PRODUCTID","PRICE","NAME"  
AS  
"CATEGORYNAME"  
from :product where  
"PRICE">50;  

## 9.CE\_CALC

这个函数用于其它关系操作的函数内部，通常是计算一个表达式的值并且绑定为一个新列，比较常用于CE\_PROJECTION函数中。他一般有两个参数，分别为计算的表达式和结果的数据类型。计算的表达式中还可以使用一些函数，包括转换函数、字符串函数、数学函数、日期函数以及一些其它函数。  
例如：选取表变量:product中的cid, cname, oid, sales四列，并将sales与变量:vat\_rate所得的乘积作为新的一列，重命名为” SALES\_VAT”。with\_tax = CE\_PROJECTION(:product, ["CID", "CNAME", "OID", "SALES",CE\_CALC('"SALES" \* :vat\_rate',decimal(10,2)) AS"SALES\_VAT"]);    
  
     使用SQL语句表示为：  
  
with\_tax2 = SELECT cid, cname, oid, sales,sales \* :vat\_rate as sales\_vat FROM :product;  

## 10.CE\_AGGREGATION

这个函数对输入列分组并且计算每个分组的聚合结果，包含常用的COUNT 、 SUM 、 MIN 、 MAX聚合函数。  
     例如：对与表变量:product按year进行分组，并计算每组的count()值。  
  
product5 = CE\_AGGREGATION (:product,[COUNT ("PRODUCT\_ID") AS"CNT"], ["YEAR"]);    
  
     使用SQL语句表示为：  
  
product5 = SELECT  
COUNT ("PRODUCT\_ID") AS cnt, year  
FROM :product GROUP  
BY  
year;  

## 11.CE\_UNION\_ALL

这个函数计算两个具有相同列名和列顺序表变量UNION ALL的操作，即存在相同行均保留。  
     例如：求两个表变量:product1和:product2的UNION ALL 操作，必须保证俩个变量的列名和顺序均相同。  
  
              pro\_union\_all = CE\_UNION\_ALL (:product1, :product2);    
  
     使用SQL语句表示为：  
  
pro\_union\_all = SELECT \* FROM :product1 UNIONALLSELECT \* FROM :product2;    
其他CE函数

## 12.CE\_VERTICAL\_UNION

这个函数的功能是实现垂直UNION，即在保证行数相同的条件下，选取两个表变量的全部或者部分列拼接成为一个表变量。  
     例如：将表变量:firstname的"ID", "FIRSTNAME"两列和表变量:lastname的"LASTNAME"列拼接成为一个新的表变量。  
  
  
out = CE\_VERTICAL\_UNION(:firstname, ["ID", "FIRSTNAME"  
AS  
"GIVENNAME"], :lastname, ["LASTNAME"  
AS  
"FAMILYNAME"]);    
  
特殊使用1.加编号  
在使用CE\_CALC()时，可以使用rownum()这个函数，返回编号为新的一列，因此可以使用此方式加上编号列。  
例如：  
  
product2 = CE\_PROJECTION(:product1, ["NAME","COLOR","PRICE",CE\_CALC('rownum()',integer) as  
"ROWNUM"]);    
  
     所得到的结果即2.取前n项  
     根据上一例子的使用，引申类似的使用。根据rownum()得到编号后，即可以选取前n项。  
     例如：取表变量：product1的前五列

product3 = CE\_PROJECTION(:product1, ["NAME","COLOR","PRICE",CE\_CALC('rownum()',integer) as"ROWNUM"],'"ROWNUM"<5');    
  
     所得到的结果即使用注意使用CE function由于直接由calculation engine处理，效率相对较高，可能会带来性能的提升。最好不要把CE function和普通的SQL语句混合使用，因为他们运行在不同的engine上，要避免数据在不同的引擎中反复传递可能会带来性能上的问题。  
使用限制现有的CE function并不能实现所有常用的SQL查询所能完成的操作。例如连接操作，只有CE\_UNION\_ALL,可以实现UNION  ALL的操作，而没有与UNION直接相对应的CE function。同样，对于连接操作，没有于外连接(FULL OUTER JOIN)相对应的CE function。但是，可以使用其他CE function结合使用，或使用其他方式来实现相应的功能。

关于CE function的使用，请参考SAP HANA SQLScript Reference