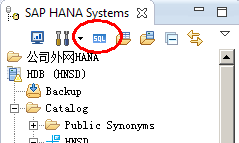
HANA存储过程

存储过程是可以重用的处理模块，在HANA中，可以采用SQLScript、L或者是R语言进行执行

在HANA中，有两种创建PROCEDURE的方式，一种是用SQL editor进行创建，一种是用package中的创建向导进行创建，这两种方式创建的PROC在保存方式、调用、参数上有一些不同，将在相关内容一一对比。

1. 创建
   1. SQL editor

* 按下图打开SQL CONSOLE



存储过程的创建语法为

|  |
| --- |
| **CREATE PROCEDURE {schema.}name**  **{({IN|OUT|INOUT}**  **param\_name data\_type {,...})}**  **{LANGUAGE <LANG>} {SQL SECURITY <MODE>}**  **{READS SQL DATA {WITH RESULT VIEW <view\_name>}} AS**  **BEGIN**  **...**  **END**  **来源于《HA300\_EN\_Col97\_FV\_Show.pdf》第150页** |

<LANG>是指存储过程执行的语句，主要是SQLScript、L或者是R语言，如果不指定，则默认SQLscript

<MODE>是指definer 和 invoker，及定义者权限和调用者权限

READS SQL DATA定义存储过程为read-only

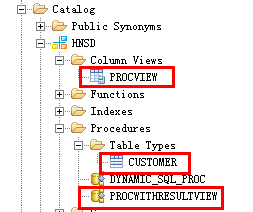
WITH RESULT VIEW是用于创建一个可用于sql语句的column view，其数据集内容为存储过程的表类型输出参数所包含的数据集。

以下是创建实例

|  |
| --- |
| **/\*创建表类型**CUSTOMER **\*/**  **CREATE** **TYPE** "HNSD"."CUSTOMER" **AS** **TABLE** ( "CODE" INT CS\_INT )  **/\*创建存储过程**ProcWithResultView**，输出视图为**ProcView **\*/**  **CREATE** **PROCEDURE** ProcWithResultView(**IN** id INT, **OUT** o1 CUSTOMER)  **LANGUAGE** SQLSCRIPT READS **SQL** DATA  **WITH** RESULT **VIEW** ProcView  **AS**  **BEGIN**  o1 = **SELECT** :id **as** CODE **FROM** DUMMY;  **END**; |

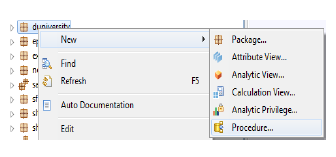
需要注意的是输入参数可以为基本类型和自定义table类型，输出参数可以为基本类型和自定义table类型，如果定义了resultView，则输出参数必须是table类型，并且该table类型已经定义完成。

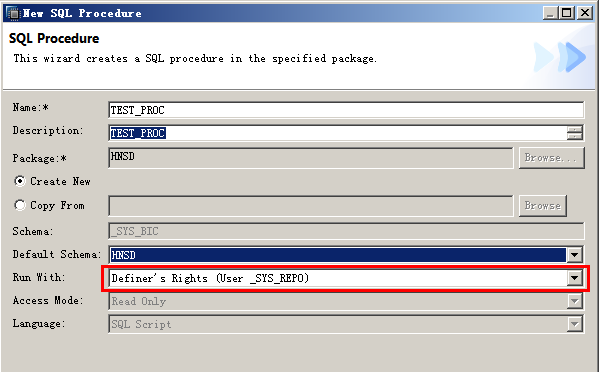
执行创建sql后，存储过程将保存目录如下



* 1. 创建向导

按下图打开存储过程创建向导





如上图所示，在创建页面有如下属性

Name：存储过程名称

Description：存储过程描述

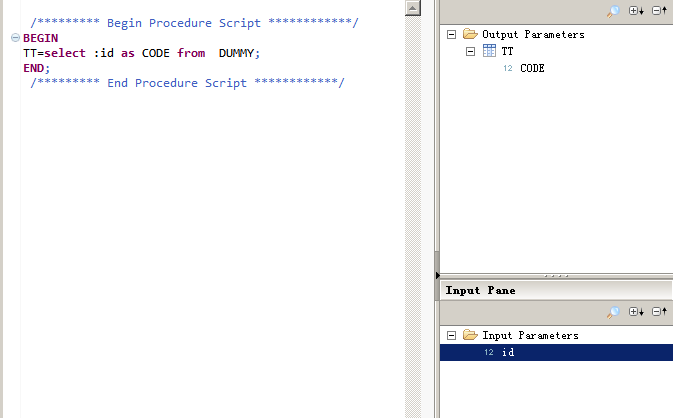
Default Schema：默认的Schema名称，建议选择用户所属的Schema，这个关系着在\_SYS\_BIC中的存储过程名称

Run With：Definer 定义者权限，Invoker 调用者权限，定义者权限将会限制其他用户调用该存储过程，调用者权限将允许系统向其他用户授权调用该存储过程。

Access Mode：Read Only，Read-Wirter，但是在现使用studio中不可选

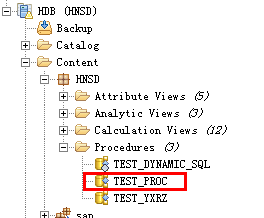
Language：SQL Script，L，R，但是在现使用studio中不可选

如下图，为存储过程的脚本编辑视图

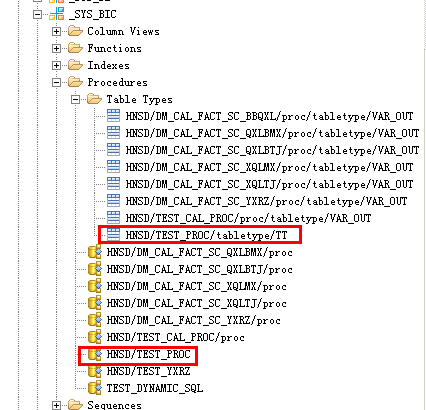


在向导创建存储过程中，输入参数可以为基本类型和自定义table类型，而输出参数必须为自定义table类型，并且该table类型无需预先定义，而是在参数定义栏中进行设置，在存储过程save and validate过程中自动创建

创建完成后，保存目录如下

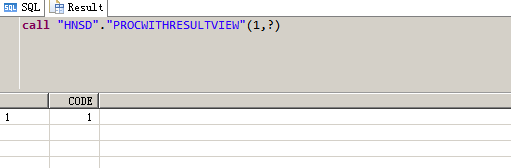


同时在\_SYS\_BIC下会生成如下对象



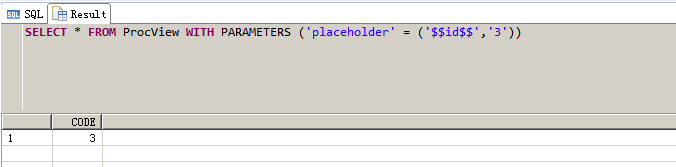
1. 调用
   1. SQL editor

在SQL CONSOLE中调用方式如下

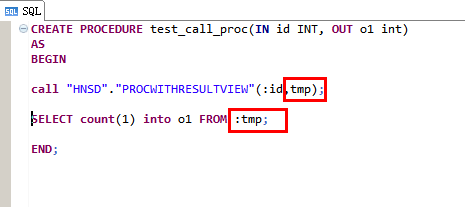


另外，如果定义了result view，可以直接在sql语句中调用

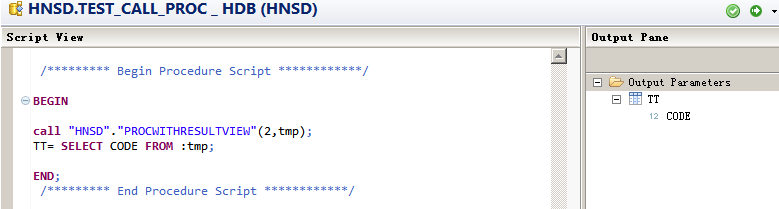
**SELECT** \* **FROM** ProcView **WITH** PARAMETERS ('placeholder' = ('$$id$$','3'))



在存储过程中调用方式如下



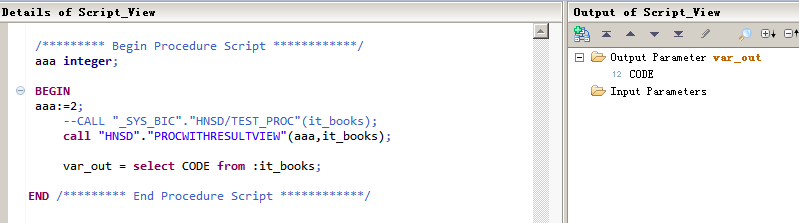
tmp是存储过程中使用的临时参数，用于接收调用存储过程的输出结果集，然后该结果集可以在sql中直接调用。



该方式在save and validate中报错，错误显示权限不足，原因尚未明确



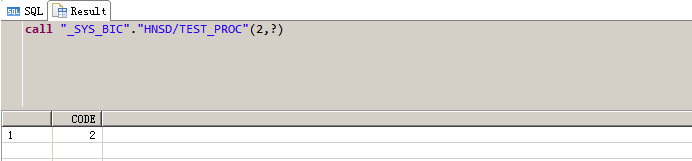
在计算视图中调用



由上可以看出，计算视图实际上和存储过程类似，语法和结构都比较相近。

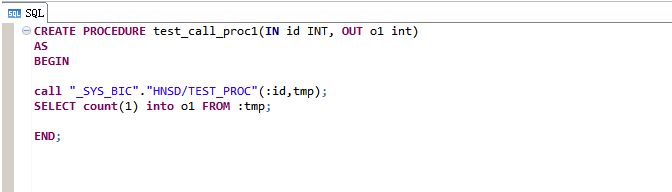
* 1. 创建向导

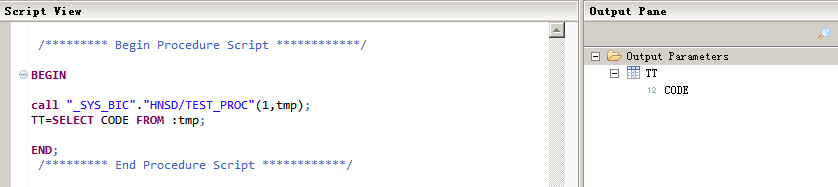
在SQL CONSOLE中的调用方式如下



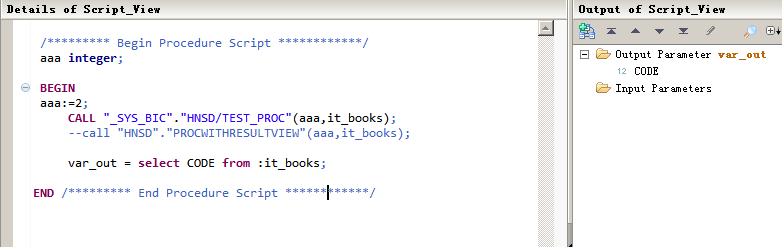
在向导中创建的存储过程，无法直接被调用，只能是调用\_SYS\_BIC中的生成对象，同时因为无法定义result view，所以无法直接在sql中被调用

在存储过程中调用方式如下





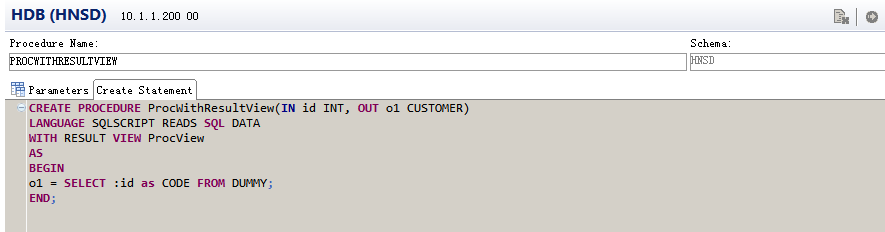
在计算视图中调用



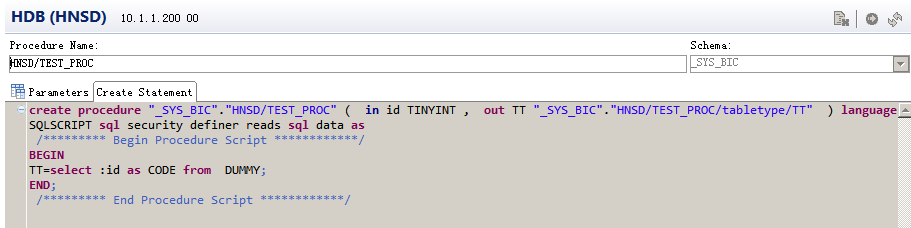
1. 两种方式的区别和相同

* 我们从存储过程的Create Statement来分析这两种方式的相同和不同之处

SQL Editor



创建向导

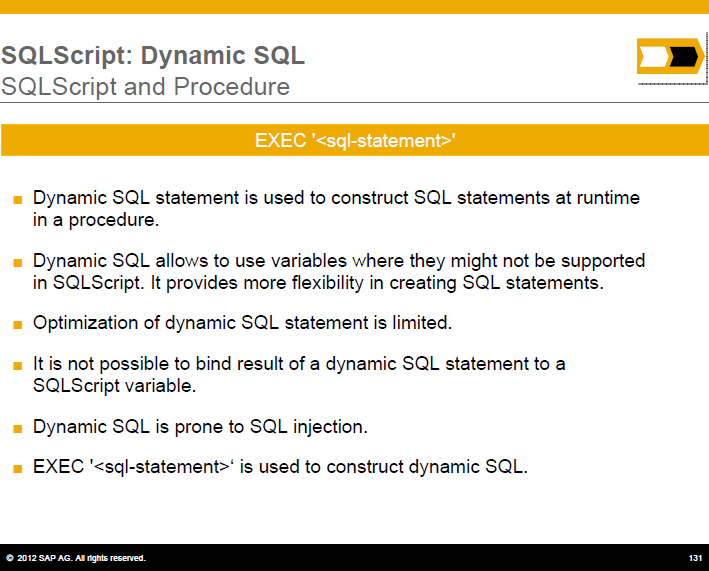


从以上创建语句来看，两种方式的存储过程没有区别，脚本语法都是一致的。

从存储过程创建方式来分析，对比如下

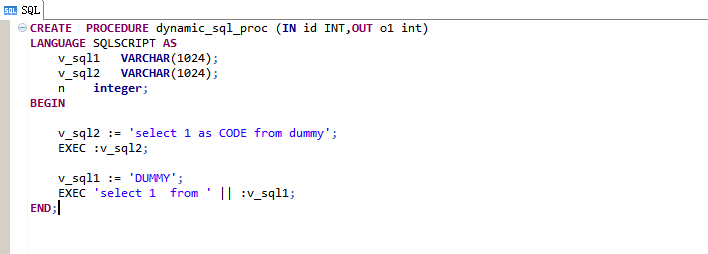
|  |  |  |
| --- | --- | --- |
| **功能** | **SQL Editor** | **创建向导** |
| **输入参数** | 可以为基本类型、自定义TABLE类型 | 相同 |
| **输出参数** | 可以为基本类型、自定义TABLE类型 | 只能为自定义TABLE类型 |
| **LANGUAGE** | SQLSCRIPT，L，R | 相同 |
| **SQL SECURITY** | definer，invoker | 相同 |
| **READS SQL DATA** | Read only | 相同 |
| **WITH RESULT VIEW** | 可以定义column view | 没法定义 |
| **修改** | 没法修改，只能drop、create | 可以修改后重新active |

1. 动态SQL
   1. 动态SQL介绍



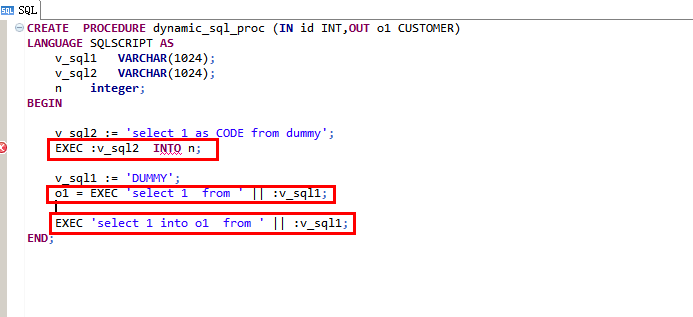
**来源于《HA300\_EN\_Col97\_FV\_Show.pdf》第171页**

* 1. 动态SQL语法

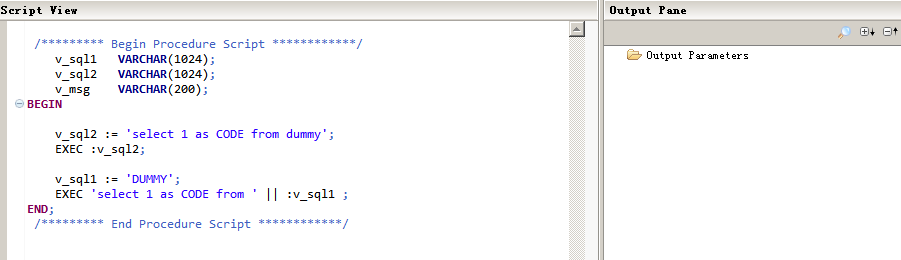


* 1. 动态SQL相关

根据介绍，动态SQL的执行结果无法与SQLScript变量绑定，也就是说如下语句是不合法的



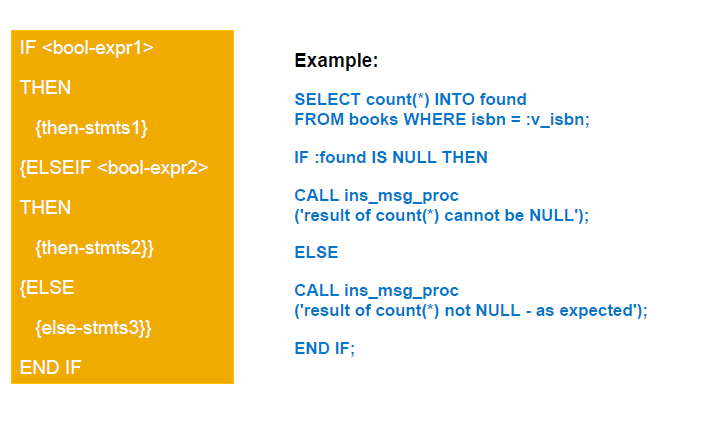
另外，在创建向导所创的存储过程中，似乎无法使用动态SQL，如下面语句



报错信息为



1. 条件&循环
   1. IF



* 1. WHILE



* 1. FOR

