LOGALI

Teoría

CDS Scalar Functions

ABAP Cloud – Modelado con CDS

Contenido

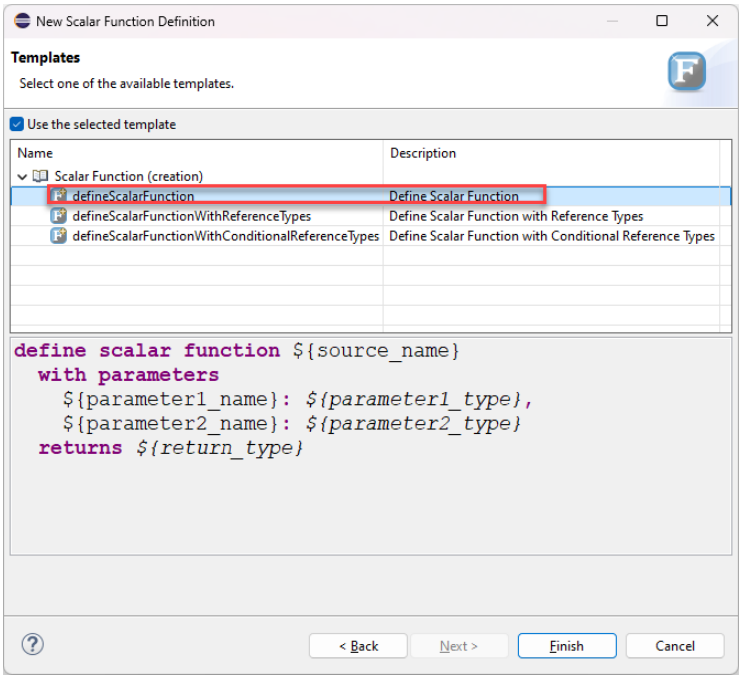
12. CDS Scalar Functions

12.1. Scalar Function – Definición

Las funciones escalares en ABAP CDS son funciones personalizadas que se utilizan para modularizar y reutilizar la lógica de negocio dentro de los modelos y vistas CDS. Estas funciones toman uno o más parámetros de entrada y devuelven un único valor escalar. A diferencia de los Table Function que devuelven una tabla o una colección de datos. Lo que las hace ideales para realizar cálculos específicos y operaciones que no requieren múltiples resultados.

La implementación de estas funciones implica definir la firma de la función, mapearla a un método AMDP que contiene la lógica de negocio, e invocar esta función desde las entidades CDS. Las funciones escalares pueden ejecutarse utilizando motores de base de datos como SQL o motores analíticos, permitiendo una integración eficiente y flexible de la lógica de negocio en el ecosistema de SAP.

Para crear un Scalar Function, se realiza a través de la carpeta de proyecto luego New en la opción **Other ABAP Repository Object** ubicar la carpeta **Core Data Services** y luego seleccionar **Scalar Function Definition** y seleccionar la plantilla **defineScalarFunction** que se encuentra en la carpeta **Scalar Function (creation).**



Sintaxis:

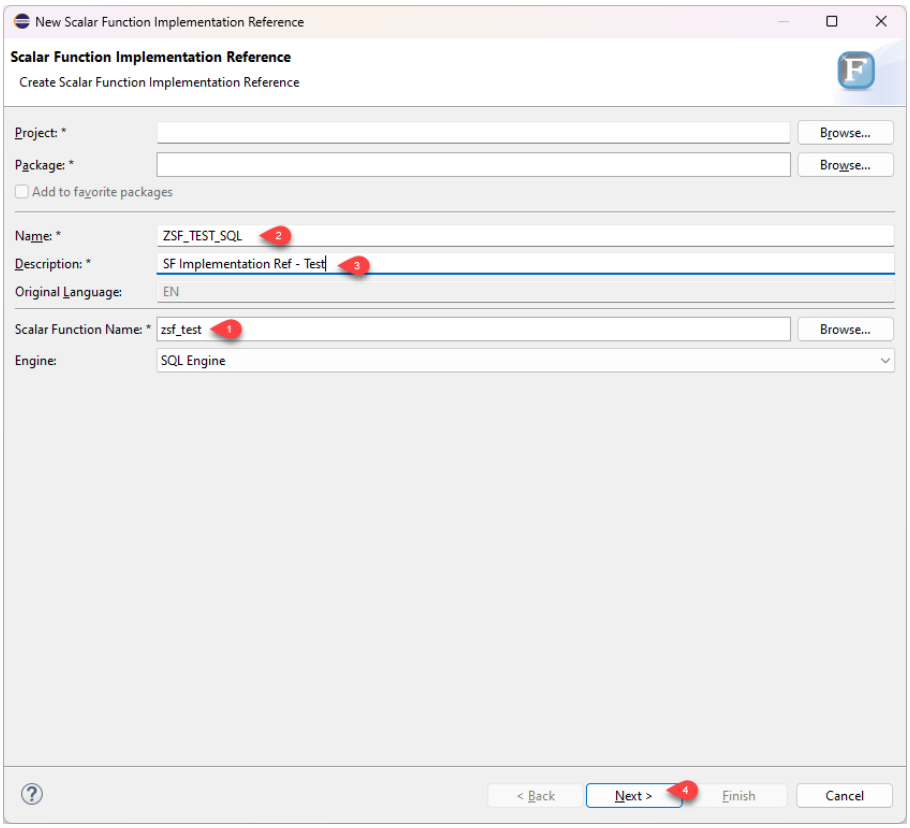
```
define scalar function scalar_function_name
 with parameters
   parameter1_name: parameter1_type,
   parameter2_name: parameter2_type
 returns return_type
```

12.2. Scalar Function – Referencia de Implementación

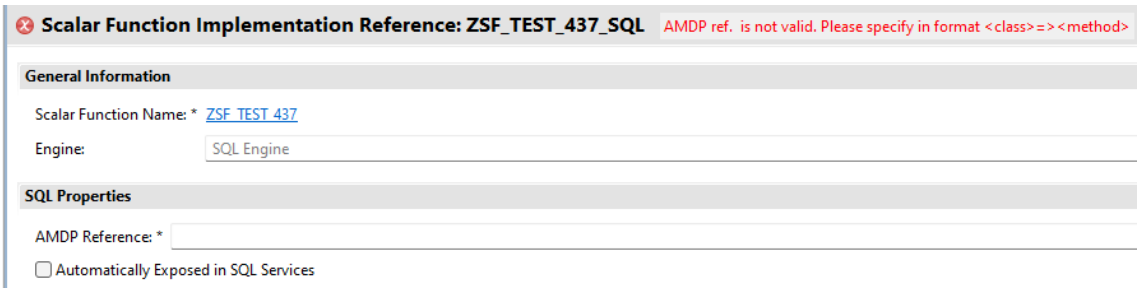
La referencia de implementación de funciones escalares en ABAP Core Data Services (CDS) es un objeto esencial que vincula la definición de una función escalar con la clase ABAP que implementará de manera obligatoria la interfaz **if_amdp_marker_hdb.** Para permitir de esta forma la definición de los métodos ABAP Managed Database Procedure (AMDP).

Para crear una implementación de referencia para una Scalar Function, se realiza a través de la carpeta de proyecto luego New en la opción **Other ABAP Repository Object** ubicar la carpeta **Core Data Services** y luego seleccionar **Scalar Function Implementation Reference.** Luego seleccionar el nombre del Scalar Function a utilizar.

Es importante señalar que la nomenclatura del nombre de la referencia de implementación se debe agregar el sufijo **_ANA** o **_SQL**. Dependiendo del motor de base de datos seleccionado Analytical Engine ó SQL Engine respectivamente.



Luego se habrá realizado la implementación de referencia para una Scalar Function. Y se mostrará de la siguiente forma:



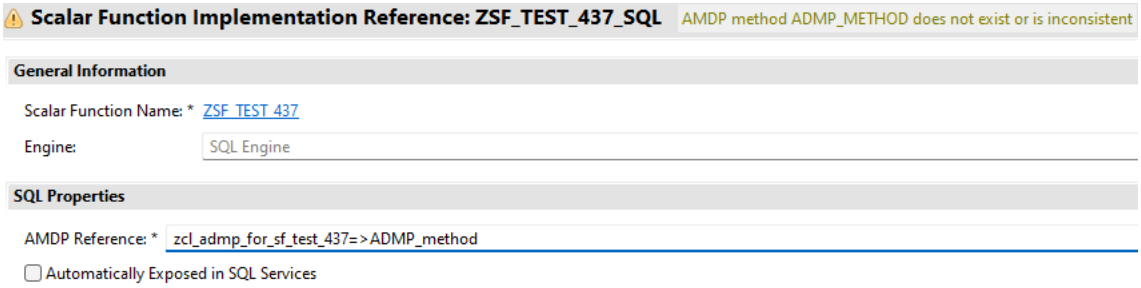
Donde es necesario agregar un nombre de la clase e implementar de manera obligatoria la interfaz **if_amdp_marker_hdb** para poder definir métodos ADMP para devolver un valor escalar.

Ejemplo de la declaración de la clase implementando la interfaz if_amdp_marker_hdb:

```
CLASS class_name DEFINITION
 PUBLIC
 FINAL
 CREATE PUBLIC .
 PUBLIC SECTION.
  INTERFACES if_amdp_marker_hdb .
```

```
 PROTECTED SECTION.
 PRIVATE SECTION.
ENDCLASS.
CLASS class_name IMPLEMENTATION.
ENDCLASS.
```

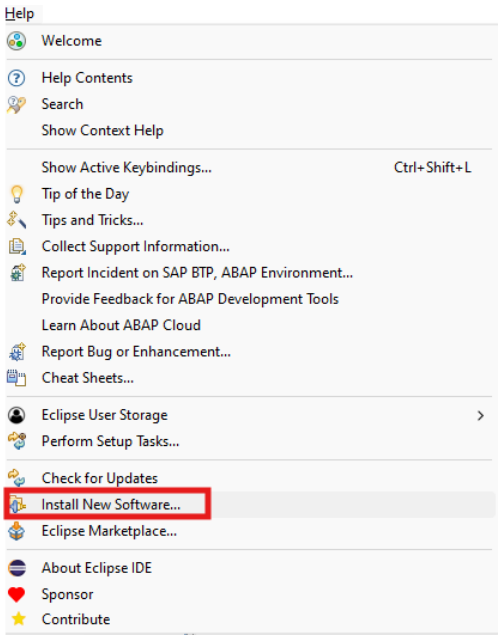
A pesar de que no se haya declarado el método dentro de la clase especificada en el campo ADMP Reference, el editor permitirá activar la implementación de referencia.



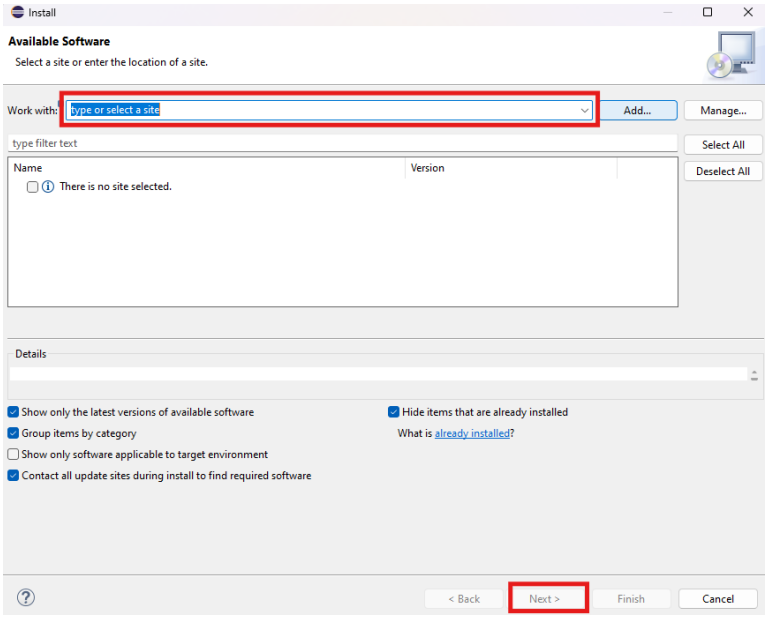
Es importante tener en cuenta de que tanto Eclipse como las herramientas de desarrollo ABAP Development Tools (ADT) estén actualizados. De lo contrario, siempre aparecerá un mensaje indicando que están desactualizados, lo que impedirá continuar con el desarrollo de la implementación de referencia para una Scalar Function. En ese caso, utilizar los siguientes enlaces para las diferentes actualizaciones:

ADT	https://tools.hana.ondemand.com/latest
Eclipse	http://download.eclipse.org/releases/latest

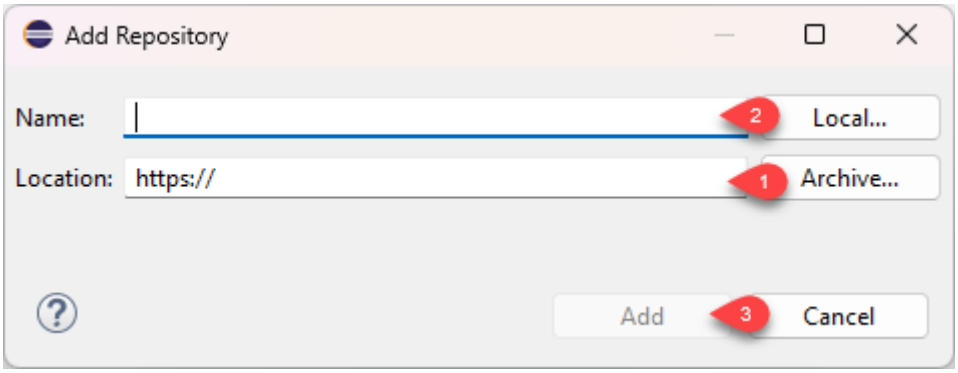
Recordando que para instalar actualizaciones se debe de ir a la pestaña **Help**, posicionada en la barra de herramientas superior del Eclipse y ubicar la opción Install New Software.



Donde se debe escribir los enlaces compartidos anteriormente en el campo **Work with.** Luego presionar enter y al cargar los componentes a actualizar presionar **Next.** Para continuar con el proceso.



Otra forma sería presionar el botón **Add…** y colocar en el campo **Name:** cualquier nombre referente con los paquetes a actualizar y colocar el enlace correspondiente en el campo **Location.** Para seguir con el proceso explicado anteriormente.



En el peor de los casos se deberá reinstalar el Eclipse y las herramientas ADT.

12.3. Scalar Function – Implementación AMDP

La implementación de funciones escalares con métodos ABAP Managed Database Procedure (AMDP) en ABAP Core Data Services (CDS) permite modularizar y reutilizar la lógica de negocio de manera eficiente. Estas funciones, que aceptan uno o más parámetros de entrada y devuelven un único valor escalar, se definen y enlazan con métodos AMDP dentro de clases ABAP. Esto optimiza la ejecución de lógica específica directamente en la base de datos, mejorando así el rendimiento y la flexibilidad del desarrollo. La implementación incluye la creación de la función escalar, su referencia de implementación y la definición del método AMDP que ejecutará la lógica.

Sintaxis para la declaración de un método AMDP:

Para declarar un método escalar, es necesario que sea un método estático de la clase y que se utilice la instrucción **FOR SCALAR FUNCTION** seguida del nombre de la función escalar correspondiente.

Sintaxis:

```
CLASS-METHODS amdp_method_for_sf FOR SCALAR FUNCTION scalar_function_name.
```

Para la implementación de dicho método escalar es necesario utilizar la instrucción **BY DATABASE FUNCTION FOR HDB LANGUAGE**

SQLSCRIPT. Para especificar que el método es una función de base de datos que se ejecutará en la base de datos HANA utilizando el lenguaje SQLScript. Donde es posible agregar la instrucción **OPTIONS READ-ONLY,** para indicar que el método sólo realizará operaciones de lectura en la base de datos y no realizará modificaciones.

Sintaxis:

```
METHOD amdp_method_for_sf BY DATABASE FUNCTION
                    FOR HDB LANGUAGE SQLSCRIPT
                    OPTIONS READ-ONLY.
 //  conditions…
result = value;

ENDMETHOD.
```

Dentro de dicho método es necesario agregar sentencia **result,** para devolver el valor escalar del método AMDP asociado a la función escalar especificada en la declaración.

Además, los parámetros declarados en la definición de una función escalar pueden ser utilizados dentro de los métodos AMDP vinculados sin necesidad de declarar variables o campos de entrada y salida correspondientes a los parámetros importing y exporting en la definición habitual de cualquier método en una clase ABAP.

Ejemplo de la implementación de un método AMDP:

```
CLASS class_name DEFINITION
 PUBLIC
 FINAL
 CREATE PUBLIC .
 PUBLIC SECTION.
   INTERFACES if_amdp_marker_hdb .
CLASS-METHODS  amdp_method_for_sf  FOR  SCALAR  FUNCTION
scalar_function_name.
 PROTECTED SECTION.
 PRIVATE SECTION.
ENDCLASS.
```

```
CLASS class_name IMPLEMENTATION.
 METHOD amdp_method_for_sf BY DATABASE FUNCTION
                    FOR HDB LANGUAGE SQLSCRIPT
                    OPTIONS READ-ONLY.
  IF parameter1_name = 'AA' then
  result = 21;
  ELSE
  result = 24;
  END if;
 ENDMETHOD.
ENDCLASS.
```

12.4. Scalar Function – Uso en entidades CDS

Las funciones escalares en ABAP Core Data Services (CDS) son funciones personalizadas utilizadas para modularizar y reutilizar la lógica de negocio en vistas y entidades CDS. Estas funciones aceptan uno o más parámetros de entrada y devuelven un único valor escalar, lo que las hace ideales para cálculos y operaciones específicas. Se integran directamente en las definiciones de CDS, optimizando el rendimiento y facilitando la creación de soluciones flexibles y eficientes en SAP HANA.

Implementación de una Función Escalar dentro de una entidad CDS:

La función escalar se integra en la sección de la declaración de los componentes de la definición de la entidad CDS utilizando su nombre y parámetros de entrada especificados. Se proyecta el resultado de la función escalar utilizando un alias para el valor de retorno. Esto permite visualizar el valor calculado en la salida de la consulta.

Sintaxis:

scalar_function_name**(**parameter1_name **=>** cds_component, … **) as** AliasName

Es importante mencionar, que el componente de la entidad utilizado en el parámetro debe ser el nombre original y no un alias para que sea válido.

Ejemplo:

Asumiendo que previamente se ha creado la función escalar junto con su implementación y la clase utilizada para devolver el valor escalar, tenemos lo siguiente:

Declaración de una entidad CDS utilizando funciones escalares:

```
@AbapCatalog.viewEnhancementCategory: [#NONE]
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'CDS - CDS with Scalar Functions'
@Metadata.ignorePropagatedAnnotations: true
@ObjectModel.usageType:{
  serviceQuality: #X,
  sizeCategory: #S,
  dataClass: #MIXED
}
define view entity cds_entity_name
 as select from data_source_name
{
 key key_component            as AliasName,
   component                  as AliasName,
scalar_function_name(parameter1_name => cds_component ) as
AliasName

}
```

Beneficio de uso de funciones escalares en entidades CDS:

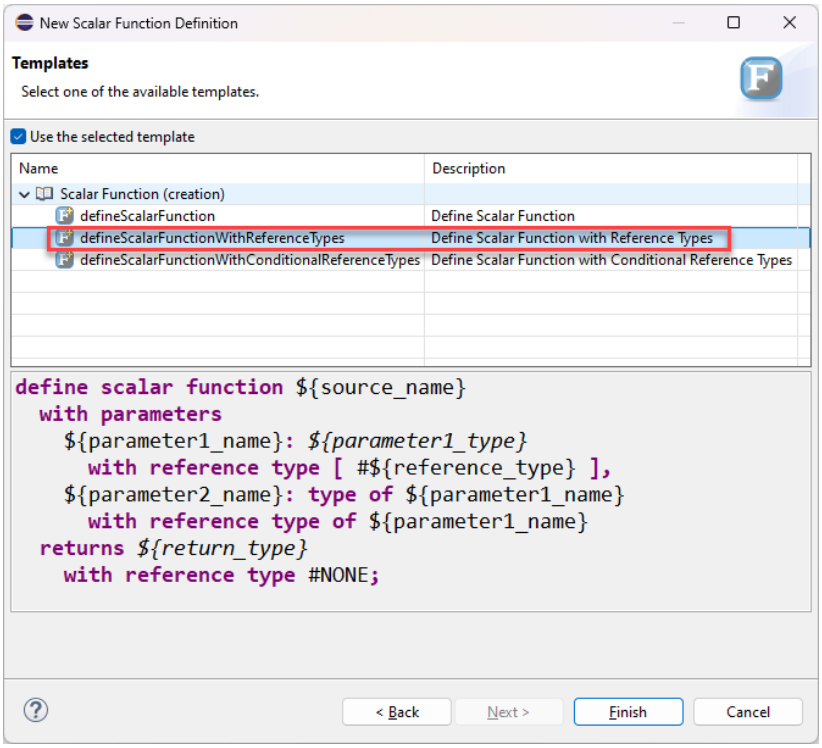
- Las funciones escalares permiten integrar cálculos específicos dentro de las vistas CDS, facilitando la modularización de la lógica de negocio.

- Estas funciones pueden reutilizarse en múltiples entidades CDS, proporcionando una solución eficiente y mantenible.
- Al ejecutar la lógica directamente en la base de datos mediante funciones escalares, se optimiza el rendimiento y se reduce la necesidad de transferir grandes volúmenes de datos entre la base de datos y la aplicación.
- Las funciones escalares pueden manejar tanto lógica simple como lógica compleja, lo que las hace muy flexibles.
- La lógica implementada puede adaptarse a los requisitos específicos del negocio, asegurando que las soluciones sean adecuadas para diferentes escenarios.

12.5. Scalar Function con Tipo Referenciado

Las funciones escalares basadas en SQL son capaces de manejar tanto los campos de importe como los campos de cantidad definidos en CDS. Estos campos están referenciados a una clave de moneda, una clave de unidad o una unidad calculada. Las funciones escalares pueden gestionar estas referencias de manera eficiente. Es posible definir qué tipos de referencia son permitidos para cada parámetro de entrada y para el parámetro de retorno. Si los tipos de referencia utilizados en los parámetros de entrada no están permitidos explícitamente, se producirá un error de comprobación de sintaxis.

Para crear un Scalar Function con referencia de tipo, se realiza a través de la carpeta de proyecto luego New en la opción **Other ABAP Repository Object** ubicar la carpeta **Core Data Services** y luego seleccionar **Scalar Function Definition** y seleccionar la plantilla **defineScalarFunctionWithReferenceTypes** que se encuentra en la carpeta **Scalar Function (creation).**



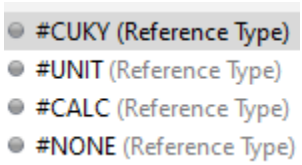
Sintaxis:

```
define scalar function scalar_function_name
 with parameters
  parameter1_name: parameter1_type
   with reference type [ #reference_type ],
  parameter2_name: type of parameter1_name
   with reference type of parameter1_name
 returns return_type
  with reference type #NONE;
```

Para establecer el tipo de referencia de un parámetro o un campo de retorno escalar, se utiliza la instrucción **WITH REFERENCE TYPE #reference_type,** seguida del tipo de referencia. Este tipo de referencia también puede especificarse directamente como una colección con un solo tipo dentro de corchetes, por ejemplo: **WITH REFERENCE TYPE [#reference_type]**.

Donde es posible colocar un tipo de referencia **#reference_type,** tanto para cada parámetro como en el valor que devuelve la función escalar. Los tipos de referencia disponibles son:

- **#CUCKY:** Se refiere a un tipo de dato que representa valores monetarios, como importes y cantidades en diferentes monedas.
- **#UNIT:** Se refiere a un tipo de dato que representa unidades de medida, como longitud, peso, volumen, etc.
- **#CALC:** Se refiere a un tipo de dato que representa valores calculados, como resultados de cálculos matemáticos.
- **#NONE:** Indica que no se especifica ningún tipo de referencia particular para el parámetro.



Como dato adicional es posible colocar directamente el nombre de un parámetro que tenga un tipo de referencia previamente configurado agregando **of** al final de la instrucción **with reference type of** seguido del nombre del parámetro. De igual forma se puede establecer el tipo de un parámetro con base en otro al especificar su nombre después de los dos puntos con la instrucción **: type of** parameter_name_with_reference_type.

Como dato adicional, es posible especificar directamente el nombre de un parámetro que tenga un tipo de referencia previamente configurado añadiendo **OF** al final de la instrucción **WITH REFERENCE TYPE**, seguido del nombre del parámetro deseado. De manera similar, se puede establecer el tipo de un parámetro en base a otro al indicar su nombre después de los dos puntos con la instrucción **: TYPE OF** parameter_name_with_reference_type.

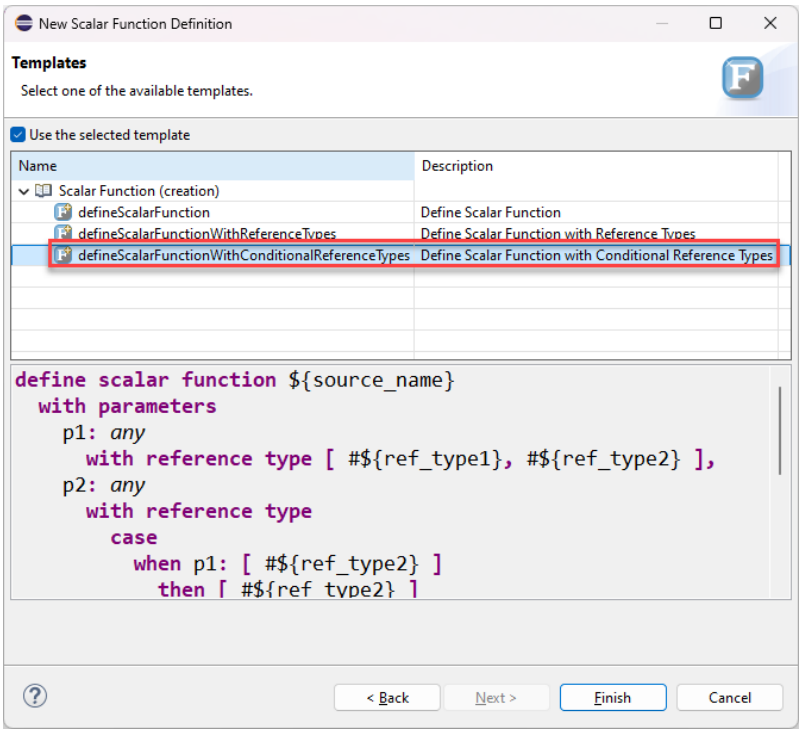
Ejemplo:

```
define scalar function scalar_function_name
 with parameters
  parameter1_name: parameter1_type,
  parameter2_name: type of parameter1_name
   with reference type of parameter1_name
 returns return_type
  with reference type of parameter2_name;
```

12.6. Scalar Function con Condición de Tipo Referenciado

El tipo de referencia también se puede definir dinámicamente, dependiendo de los tipos de referencia de los parámetros de entrada. Esto se hace mediante instrucciones **CASE** y mediante colecciones de tipos de referencia **with reference type [ref_type,...]**. Las funciones escalares con tipos referenciados condicionales en ABAP Core Data Services (CDS) son funciones que permiten definir parámetros y valores de retorno con tipos específicos, determinados en base a condiciones lógicas.

Para crear un Scalar Function con condiciones de referencia de tipos, se realiza a través de la carpeta de proyecto luego New en la opción **Other ABAP Repository Object** ubicar la carpeta **Core Data Services** y luego seleccionar **Scalar Function Definition** y seleccionar la plantilla **defineScalarFunctionWithConditionalReferenceTypes** que se encuentra en la carpeta **Scalar Function (creation).**



Puntos importantes a considerar:

- Se pueden definir tantos parámetros como se requiera donde los tipos de referencia se determinan en base a condiciones específicas como por ejemplo:
  - **p2: [#ref_type2]:** para identificar directamente por el tipo que pueda tener un parámetro en específico.
  - **p1: [reference type of p2]:** para identificar directamente por la referencia obtenida de un parámetro en específico.
  - **p1: [reference type of p2, #ref_type]:** Similar al anterior con la diferencia que identifica por un tipo de referencia posible en el parámetro deseado.
- Se pueden colocar directamente una colección con los diferentes tipos de referencia posibles para un campo con la instrucción: **with reference type [#ref_type1, #ref_type2].** Estos tipos de referencias son **#CUCKY, #UNIT, #CALC** y **#NONE** estudiados en el tema anterior.
- La sintaxis varía ligeramente, ya que en lugar de especificar el tipo del parámetro directamente, se utiliza **numeric** (válido para el motor SQL) o **any** (válido para el motor analítico). Esto permite que el contexto de la función escalar determine cuál de los tipos establecidos corresponde al tipo correcto.

Sintaxis

```
define scalar function zsf_ref_type_cond_test_437
 with parameters
  p1: numeric
   with reference type [ #ref_type1, #ref_type2 ],
  p2: numeric
   with reference type
    case
     when p1: [ #ref_type2 ]
      then [ #ref_type2 ]
     else [ #ref_type1 ]
    end
 returns return_type
  with reference type
```

```
case
  when p1: [ reference type of p2, #NONE ] and
       p2: [ #ref_type2 ]
   then [ reference type of p1 ]
  else [ #NONE ]
end
```