LOGALI

Teoría

Virtualización

ABAP Cloud – Modelado con CDS

Contenido

13. Virtualización

13.1. Object Model – Elemento Virtual

La virtualización de elementos en Core Data Services (CDS) de SAP permite definir y manipular elementos calculados que no existen físicamente en las fuentes de datos originales. Esta técnica se implementa utilizando el Saddle Service Adaptation Description Language (SADL), lo que facilita la integración de información adicional en las vistas CDS sin modificar la estructura de las tablas subyacentes. Mediante la virtualización, se pueden agregar columnas adicionales a las vistas, implementar lógica compleja en la capa ABAP y mejorar la presentación de datos en aplicaciones Fiori y servicios OData. Esta capacidad es crucial para adaptar los modelos de datos a los requisitos empresariales cambiantes, ofreciendo flexibilidad y eficiencia en el manejo y exposición de datos en el entorno SAP.

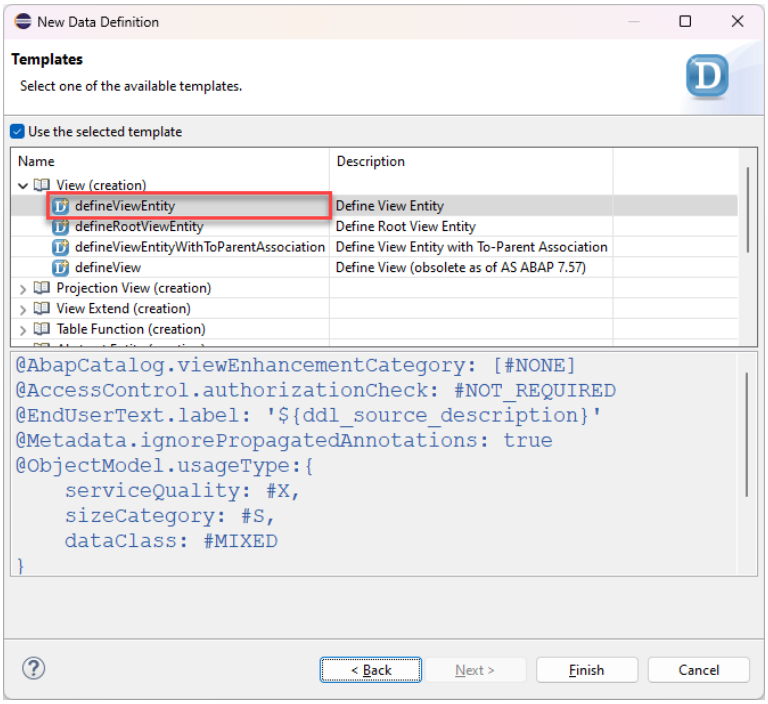
Implementación de virtualización de elementos:

Los elementos virtuales pueden ser utilizados en diferentes capas de consumo, como interfaces de usuario (UI) basadas en Fiori Elements, o en servicios OData. Permiten mostrar información adicional en las aplicaciones sin necesidad de modificar las tablas de base de datos subyacentes. Para indicar que el componente de una entidad CDS es un elemento virtual se utiliza la anotación:

```
@ObjectModel.virtualElementCalculatedBy: 'ABAP:class_name'

component as Alias_name
```

Recordando que para crear una entidad o vista CDS, 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 **Data Definition** y seleccionar la plantilla **defineViewEntity** que se encuentra en la carpeta **View (creation).**



Ejemplo:

```
@AbapCatalog.viewEnhancementCategory: [#NONE]
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'CDS - CDS with virtualization'
@Metadata.ignorePropagatedAnnotations: true
@ObjectModel.usageType:{
  serviceQuality: #X,
  sizeCategory: #S,
  dataClass: #MIXED
}
define view entity entity_name
 as select from data_source_name
{
 key component              as AliasName,
    component             as AliasName,
    @EndUserText.label: 'Label name'
    @ObjectModel.virtualElementCalculatedBy: 'ABAP:class_name'
    cast( 0 as abap.dec(16,2)) as AliasName
}
```

Puntos importantes a considerar:

- En la sección: ABAP:**class_name** se debe especificar un nombre de la clase que definirá los métodos necesarios para calcular y devolver los valores de los elementos virtuales. Dicha clase necesita implementar de manera obligatoria la **IF_SADL_EXIT_CALC_ELEMENT_READ**. Esta interfaz permite definir la lógica necesaria para calcular los valores de los elementos virtuales en la capa del servidor de aplicaciones. La entidad CDS con un elemento virtual al que se le haya especificado una clase existente permitirá ser activada.
- Además es necesario a través de un **cast** especificar el tipo de datos del componente con la virtualización.
- Es posible agregar un texto descriptivo a los elementos del tipo a través de la anotación **@EndUserText.label: 'label'.**

Declaración de la clase con la lógica de los elementos virtuales:

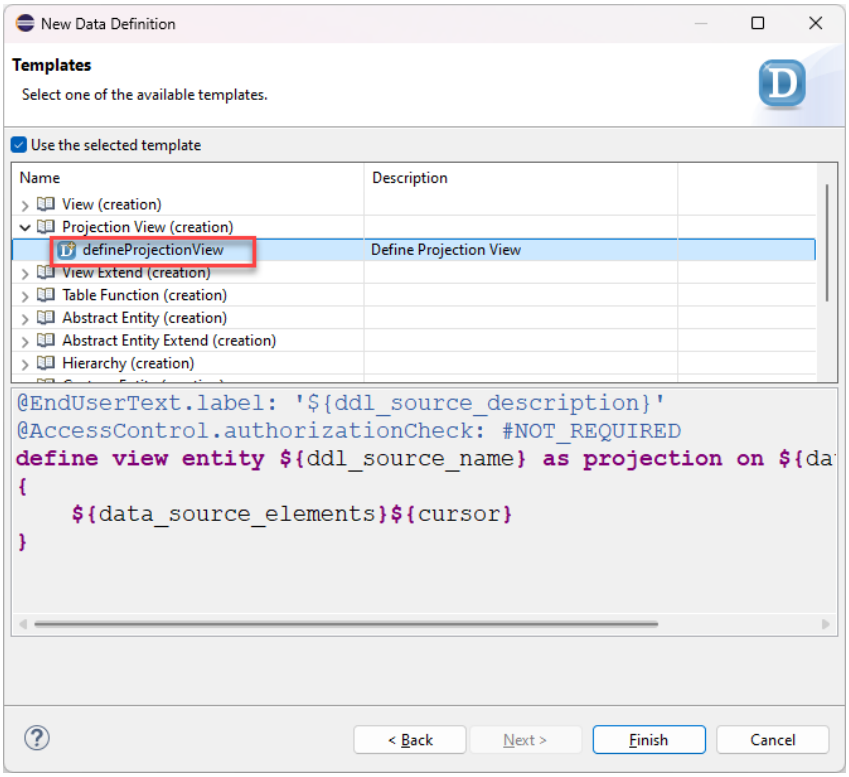
```
CLASS class_name DEFINITION
 PUBLIC
 FINAL
 CREATE PUBLIC .
 PUBLIC SECTION.
   INTERFACES if_sadl_exit_calc_element_read.
 PROTECTED SECTION.
 PRIVATE SECTION.
ENDCLASS.
CLASS class_name IMPLEMENTATION.
 METHOD if_sadl_exit_calc_element_read~calculate.
 ENDMETHOD.
 METHOD if_sadl_exit_calc_element_read~get_calculation_info.
 ENDMETHOD.
ENDCLASS.
```

Vistas de proyección:

Para definir una virtualización de elementos en las vistas de proyección es necesario cambiar un poco la sintaxis, ya que la forma tradicional no es compatible la diferencia es que es necesario colocar

la instrucción **virtual,** seguido del nombre y su tipo junto con la anotación **@ObjectModel.virtualElementCalculatedBy.**

Recordando que para crear una proyección se realiza el mismo procedimiento para crear una entidad o vista CDS 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 **Data Definition** y seleccionar la plantilla **defineProjectionView** que se encuentra en la carpeta **Projection View (creation).**



Además, es necesario agregar la instrucción root para convertirla en una entidad raíz, lo que requiere que exista una entidad raíz CDS en la fuente de datos de la proyección. También es esencial que la proyección sea del tipo de contrato **transactional_query**, para permitir que la vista de proyección CDS exponga datos de manera adecuada para operaciones transaccionales.

Ejemplo:

```
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'CDS - CDS with virtualization on projection'
@Metadata.ignorePropagatedAnnotations: true
define root view entity entity_name
 provider contract transactional_query
```

```
 as projection on cds_root_source
{
 key component              as AliasName,
    component             as AliasName,
    @EndUserText.label: 'Label name'
    @ObjectModel.virtualElementCalculatedBy: 'ABAP:class_name'
 virtual component : abap.dec( 16, 2 )
}
```

13.2. SADL Exit – Implementación

La implementación de SADL Exit (Service Adaptation Description Language Exit) en CDS (Core Data Services) es una técnica avanzada que permite definir y gestionar elementos virtuales dentro de las entidades CDS. Esta técnica habilita la extensión de las vistas CDS mediante la adición de columnas calculadas que no existen en las fuentes de datos originales. A través de SADL Exit, es posible integrar lógica compleja en la capa ABAP, permitiendo la exposición de datos adicionales de manera dinámica en servicios OData y aplicaciones Fiori, sin necesidad de modificar las tablas subyacentes en la base de datos. Esta funcionalidad ofrece una mayor flexibilidad y adaptabilidad para satisfacer los requerimientos empresariales específicos.

Métodos SADL Exit:

Los métodos SADL Exit son fundamentales cuando es requerido extender la funcionalidad de las vistas CDS al incluir elementos virtuales, es decir, componentes que no existen físicamente en las tablas de base de datos, pero que son calculadas en tiempo de ejecución. Estos métodos son los siguientes:

- **get_calculation_info:** Este método se utiliza en el contexto de la virtualización de elementos en CDS para especificar qué información adicional se necesita para calcular los valores de los elementos virtuales. Y posee los siguientes parámetros:
  - **iv_entity**: Nombre de la entidad CDS.
  - **it_requested_calc_elements:** elementos a los cuales se le hace referencia.

- **et_requested_orig_elements:** y se devuelven los datos en esta estructura.

- **calculate:** Este método tiene la función principal de realizar el cálculo real de los valores de los elementos virtuales basándose en la información solicitada previamente por el método **get_calculation_info**. Este método es fundamental para la virtualización de elementos en CDS y se encarga de llenar los elementos virtuales con los valores calculados. Y posee los siguientes parámetros:
  - **it_original_data:** Tabla de datos originales, al menos completa para los elementos originales solicitados.
  - **it_requested_calc_elements:** Elementos de cálculo solicitados (transitorios).
  - **ct_calculated_data:** Tabla de campos calculados con correspondencia 1:1 con los datos originales por índice.

Ejemplo:

```
CLASS class_name DEFINITION
 PUBLIC
 FINAL
 CREATE PUBLIC .
 PUBLIC SECTION.
  INTERFACES if_sadl_exit_calc_element_read.
 PROTECTED SECTION.
 PRIVATE SECTION.
ENDCLASS.
CLASS class_name IMPLEMENTATION.
 METHOD if_sadl_exit_calc_element_read~calculate.
 DATA lt_original_table TYPE STANDARD TABLE OF
cds_entity_name WITH DEFAULT KEY.
      lt_original_table = CORRESPONDING #( it_original_data ).
LOOP AT lt_original_table ASSIGNING FIELD-
SYMBOL(<fs_original_data>).
    <fs_original_data>-Component = <fs_original_data>-Component
- ( <fs_original_data>-Component * ( 1 / 10 ) ).
  ENDLOOP.
  ct_calculated_data = CORRESPONDING #( lt_original_table ).
```

```
 ENDMETHOD.
 METHOD if_sadl_exit_calc_element_read~get_calculation_info.
  CASE iv_entity.
   WHEN 'cds_entity_name_1' OR 'cds_entity_name_2' .
    LOOP AT it_requested_calc_elements INTO
DATA(ls_requested_calc_elements).
      APPEND 'Component' TO et_requested_orig_elements.
    ENDLOOP.
  ENDCASE.
 ENDMETHOD.
ENDCLASS.
```

Ventajas de la Virtualización mediante SADL:

- **Flexibilidad:** Permite adaptar rápidamente las vistas CDS a nuevos requerimientos de negocio sin modificar las estructuras de datos subyacentes.
- **Reutilización de Lógica:** Centraliza la lógica de cálculo en la capa ABAP, facilitando su mantenimiento y reutilización en diferentes vistas y aplicaciones.
- **Optimización del Rendimiento:** Desplaza la carga de cálculos complejos a la capa del servidor de aplicaciones, aprovechando las capacidades de procesamiento de SAP HANA para operaciones SQL intensivas.

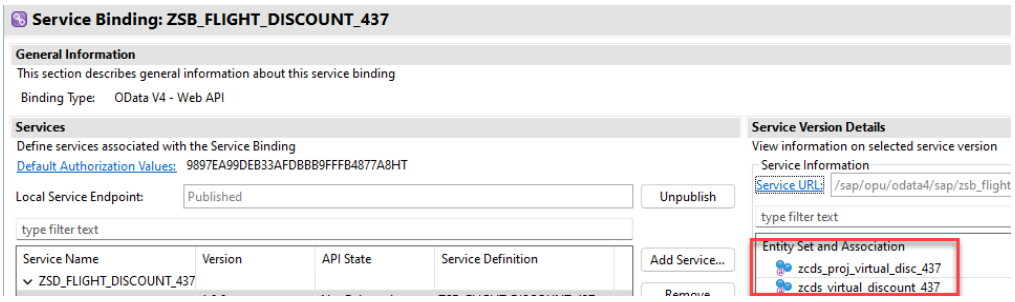
Consideraciones y Limitaciones:

- **Contexto de Ejecución:** La virtualización mediante SADL solo se ejecuta en el contexto de llamadas HTTP de tipo OData, no en operaciones SQL directas en la base de datos. Por lo tanto, al consultar los registros de una entidad CDS con un elemento con virtualización mediante operaciones SQL o a través de la herramienta Data Preview, los cálculos configurados a ese componente no podrán ser mostrados.
- **Impacto en el Rendimiento:** Aunque se evita la sobrecarga de múltiples llamadas, el procesamiento adicional en la capa ABAP puede afectar el tiempo de respuesta del servicio, especialmente con grandes volúmenes de datos.

13.3. Servicio OData – Publicación

La exposición de entidades CDS (Core Data Services) a través de servicios OData es un proceso que permite acceder y manipular datos de SAP a través de una interfaz estándar basada en HTTP. Este proceso incluye la definición de servicios, creación de bindings y la publicación de APIs, facilitando la interacción con aplicaciones externas y usuarios finales. Mediante la utilización de métodos de virtualización, como los proporcionados por SADL (Service Adaptation Description Language), se pueden calcular y exponer elementos virtuales que no existen físicamente en la base de datos, pero que son esenciales para las funcionalidades de negocio. Esta técnica proporciona flexibilidad, accesibilidad y la capacidad de adaptar rápidamente los modelos de datos a los requerimientos cambiantes del negocio.

Recordando que para publicar un servicio odata es necesario crear un service definition donde se expongan las entidades cds a utilizar y crear un service binding que utilice el service definition creado anteriormente para publicar un endpoint o url donde se pueda consultar la información de la entidad CDS con los elementos de virtualización implementados. Estos conocimientos pueden ser refrescados en la **Documentación - Tipos de Entidades y Servicios.**



Al haber expuesto las entidades en el service definition y al vincularlas con el service binding al momento de generar el servicio Odata aparecerán las entidades en la sección Entity Set and Association. Es posible agregar un alias a las entidades expuestas en el service definition, aunque si no se les coloca aparecerán con el nombre original de la entidad.

13.4. Comparación – Scalar Function | Elementos Virtuales

Las funciones escalares y los elementos virtuales son técnicas utilizadas en Core Data Services (CDS) para agregar elementos calculados a las vistas, mejorando la funcionalidad y la flexibilidad de las aplicaciones SAP. Las funciones escalares se ejecutan en la base de datos HANA y son ideales para cálculos rápidos y simples, proporcionando alto rendimiento. En cambio, los elementos virtuales se calculan en la capa ABAP utilizando la interfaz **IF_SADL_EXIT_CALC_ELEMENT_READ**, permitiendo integrar lógica de negocio más compleja y exponiendo los resultados a través de servicios OData. Cada técnica tiene su propósito específico y se elige según la complejidad de la lógica requerida y el rendimiento esperado.

Comparación y Casos de Uso Específicos:

- **Rendimiento:** Las funciones escalares ofrecen un mejor rendimiento debido a su ejecución directa en la base de datos HANA. Los elementos virtuales pueden tener un mayor impacto en el tiempo de respuesta debido a la iteración en la capa ABAP.
- **Complejidad de la Lógica:** Para lógica simple y directa, las funciones escalares son preferibles. Para lógica compleja que puede involucrar múltiples pasos o integraciones, los elementos virtuales son más adecuados.
- **Exposición de Datos:** Si los datos necesitan ser expuestos a través de servicios web, como OData, los elementos virtuales proporcionan la flexibilidad necesaria para integrar lógica avanzada.

Recomendaciones y casos de uso:

- **Uso de Funciones Escalares:** Se recomiendan para escenarios donde se requiere un cálculo simple y rápido, y donde el rendimiento es crítico. Son ideales para operaciones que

pueden ser completamente resueltas en la base de datos sin lógica adicional compleja.

- **Uso de Elementos Virtuales:** para escenarios donde la lógica de cálculo es compleja y requiere capacidades avanzadas de programación, integración con otros sistemas, o cuando se necesitan exponer estos cálculos a través de APIs o servicios OData. Son especialmente útiles cuando se necesitan exponer datos calculados a través de servicios OData. La lógica ABAP permite manejar casos complejos que el SQL Script nativo de HANA no puede resolver de manera eficiente.